

IMAV 2024

15th annual International Micro Air Vehicle Conference and Competition

September 16-20, 2024

Bristol, United Kingdom

www.imavs.org

http://www.imavs.org/



Contents

IMAV2024 1
Preface . . . 4
International IMAV Committee . . . 5
Program Committee . . . 5
Organizing Committee . . . 5
Call for Papers . . . 6
Sessions . . . 7
Keynote Speakers . . . 9
List Of Papers . . . 11
Authors . . . 12

Papers 14
Enhancing Perching Capabilities of Flapping Wing Robots with Silicone-based Electromagnetic Claws . . . 15
Flight control of an insect-like tailless flapping wing nano-robot around hover . . . 23
A Bio-Inspired Micro Air Vehicle and the Experimental Setup for Aerodynamic Testing . . . 33
Bumblebees Exhibit Adaptive Flapping Responses to Air Disturbances . . . 41
Regulatory Compliant UAS Navigation Based on a Bitmap Grid Local Path Planning Algorithm . . . 49
Robust Heading Estimation from Polarization Images by Deep Neural Networks . . . 56
Pose Estimation with a Lightweight Visual-Inertial Neural Network for Agile UAV Flights . . . 64
Evolving Spiking Neural Networks to Mimic PID Control for Autonomous Blimps . . . 72
Accelerating Swarms : Harnessing Hardware Acceleration and Parallelization in Multi-Agent... . . . 79
Adaptive Neural Network Quadrotor Trajectory Tracking Controller Tolerant to Propeller Damage . . . 87
Binary Networks and Continual Learning for Pose Estimation from a Single Aerial Image . . . 100
Feasibility Study of a Solar Powered Hybrid Micro-Aerial Vehicle . . . 109
A nonlinear autopilot for a piloted Quad-plane . . . 116
Spherical Aerial Manipulator Robot for Exploration in Complex Forest Environments . . . 124
Dexterity Assessment of Fully-Actuated UAVs . . . 131
Design of a Tapered-Wing Tiltrotor UAV for Multi-Mission Applications . . . 139
Effect of Aerodynamic Summation on Multirotor UAV Hover Performance . . . 148
Swarm based airborne wind measurement . . . 156
Landing Zone Detection for MAVs using Depth Images and Vision Transformers . . . 163
Crossing Gate Detection Using Audio Frequency Pitch Tracking . . . 171
Comparative Analysis of Cameras for ArUco Marker Recognition in Unmanned Aerial Vehicles . . . 177
Photovoltaic plant monitoring and inspection through synergic integration of UAVs and IoT . . . 185
Flying a Quadrotor with Unknown Actuators and Sensor Configuration . . . 193
Novel Flapping Wing Micro Aerial Vehicle Design Featuring a Bio-inspired Abdomen Mechanism . . . 200
A method for determining the safe separation to an unknown-model uncooperative UAV in U-space . . . 209
A Detect And Avoid concept for UAS operations near Hospital Emergency Medical Services helipads in... . . . 219
Extended Reception Coverage and Spoofing Localization of Direct Remote ID utilizing Ad-Hoc Mesh... . . . 228
Effects of Simultaneous Turbulence and Ice Accretion on Small Fixed-Wing UAV Performance . . . 236
Gen4jectory algorithm – 4-D trajectory planning with minimised flight time for multiple rotary-wing... . . . 243
Control of the DarkO Tail-Sitter Drone through an LMI-Based Static Output Feedback Design . . . 254
A Study of Gain Tuning for Prolonged Physical Contact of Fully-Actuated UAVs . . . 262
Restricted Model Control applied to the Quanser Aero with uncertain parameters . . . 270
On the Lateral Non-Stabilizability of PID Controllers for Vision-Based Line-Following Multicopters... . . . 277
Swarm-in-Blocks: Simplifying Drone Swarm Programming with Block-Based Language . . . 285

http://www.imavs.org/

Comparative analysis of ESC and Flight Controller Communication Protocols and their impact on... 293
Exploring the potential of collaborative UAV 3D mapping in Kenyan savanna for wildlife research 301
Effects of Wind Speed on Quadcopter Ceiling Effect 309
Drone Swarms for Animal Monitoring: A Method for Collecting High-Quality Multi-Perspective Data 316
WildBridge: Conservation Software for Animal Localisation using Commercial Drones 324

http://www.imavs.org/

Preface

On behalf of the Local Organizing Committee, it is our pleasure to present the proceedings of the 15th annual International Micro Air Vehicle Conference and Competition, which was held in Bristol, United Kingdom from September 16-20, 2024. For the first time, the IMAV was organized by the University of Bristol in the United Kingdom.

These proceedings are available to the public as open-access publications, seeking to promote and contribute to the advancement of the state-of-the-art in the area of small flying robots and their applications for the benefit of society.

The IMAV is a pioneering scientific-technological event in the field of aerial robotics and has been established as the primary event for the communities of researchers dedicated to the study, development, and research of Micro Air Vehicles. By participating in authentic competition scenarios, encompassing indoor and outdoor environments, as well as presenting innovative solutions via conference papers, the boundaries of research in the field of Micro Air Vehicles (MAVs) are continuously pushed forward.

These proceedings contain thirty-nine peer-reviewed scientific papers by one hundred and forty-four authors organized in seven sessions presented at the IMAV in 2024. The topics of these papers contain a nice mix ranging from aerial vehicle design and energy sources to control, navigation and perception. Together, the papers give an overview of the current state-of-the-art in the field of Micro Air Vehicles. Based on the quality of the scientific and technical contributions, papers were selected to be published in two scientific journals: the International Journal of Micro Air Vehicles (Sage), and Unmanned Systems (World Scientific).

We would like to express our sincere gratitude for the guidance and support of the members of the seventeen members of the International Committee from thirteen institutes who guide IMAV overall, and the eighteen reviewers from seven institutes, who played an important role in assuring the quality of all the papers. In addition to the presentation of the scientific papers, keynote talks were delivered by experts in the field whom we want to thank for their valuable contributions. We also want to thank all members of the Local Organizing Committee for their invaluable support in the organization of this IMAV-2024.

Bristol, United Kingdom. September 2024

T. Richardson
University of Bristol, United Kingdom

http://www.imavs.org/

International IMAV Committee

Bart Remes . . . . . TUDelft, Delft, the Netherlands
Ben M. Chen . . . . . CUHK, Hong-Kong, China
Christophe De Wagter . . . . TUDelft, Delft, the Netherlands
Dieter Moormann . . . . . RWTH, Aachen, Germany
Gautier Hattenberger . . . . . ENAC, Toulouse, France
Guido de Croon . . . . . TUDelft, Delft, the Netherlands
Hector J. Garcia de Marina Peinado . . UCM, Madrid, Spain
Jean-Marc Moschetta . . . . . ISAE, Toulouse, France
José Martinez-Carranza . . . . . INAOE, Puebla, Mexico
Leandro Lustosa . . . . . ISAE-SUPAERO, Toulouse, France
Mohamed Abdulghani . . . . . RMIT, Melbourne, Australia
Murat Bronz . . . . . ENAC, Toulouse, France
Pascual Campoy . . . . . UPM, Madrid, Spain
Renaud Kiefer . . . . . INSA, Strasbourg, France
Simon Watkins . . . . . RMIT, Melbourne, Australia
Thipyopas Chinnapat . . . . . KU, Bangkok, Thailand
Tom Richardson . . . . . UOB, Bristol, UK

Program Committee

Arthur Richards . . . . . UB, Bristol, UK
Bahadir Kocer . . . . . UB, Bristol, UK
Bas Van Oudheusden . . . . . TUDelft, Delft, the Netherlands
Christophe De Wagter . . . . TUDelft, Delft, the Netherlands
Ewoud Smeur . . . . . TUDelft, Delft, the Netherlands
Freddie Turner . . . . . UB, Bristol, UK
Gautier Hattenberger . . . . . ENAC, Toulouse, France
Guido de Croon . . . . . TUDelft, Delft, the Netherlands
José Martinez-Carranza . . . . . INAOE, Puebla, Mexico
Kilian Meier . . . . . UB, Bristol, UK
Murat Bronz . . . . . ENAC, Toulouse, France
Salua Hamaza . . . . . TUDelft, Delft, the Netherlands
Shane Windsor . . . . . UB, Bristol, UK
Simon Watkins . . . . . RMIT, Melbourne, Australia
Thipyopas Chinnapat . . . . . KU, Bangkok, Thailand
Thomas David . . . . . UB, Bristol, UK
Tom Richardson . . . . . UB, Bristol, UK
Ulrik Lundquist . . . . . SDU, Odense, Denmark

Organizing Committee

Steve Burrow (UB) . . . . . Outdoor Competition
Kjeld Jensen (SDU) . . . . . Organisation
Basaran Bahadir Kocer (UB) . . . . . Conference Chair
Ulrik Pagh Schultz Lundquist (SDU) . . . . . Keynote speaker
Fabio Remondino, (Bruno Kessler Fdn.) . . . . . Organisation
Arthur Richards (UB) . . . . . Indoor Competition
Tom Richardson (UB) . . . . . General Chair
Shane Windsor (UB) . . . . . Organisation

http://www.imavs.org/

Call for Papers



The International Micro Air Vehicles (IMAV) Conference is an annual event that brings together researchers, engineers, and enthusiasts from around the world to discuss and share the latest advances in the field of Micro Air Vehicles (MAVs). A key attraction of this event is the combination of a practical student competition with academic exchange of the latest research within a conference setting.

At the IMAV conference, attendees have the opportunity to present their research papers and attend keynote lectures from leading experts. The conference covers a wide range of topics, including the design, modeling, and control of MAVs, sensing and perception for MAVs, and applications of MAVs in various fields.

In addition to the technical presentations, the IMAV conference also features a highly anticipated flight competition, where participants compete with their autonomous MAVs in a range of tasks, such as obstacle avoidance, target detection, and delivery. The competition provides an opportunity for participants to put their research into practice and to showcase the capabilities of their MAVs in a real-world setting.

Call for papers

The call for papers for IMAV 2024 welcomed contributions to all aspects of MAVs, including, but not limited to:

- Sensing and perception for MAVs
- Multi-MAV systems and swarm intelligence
- Low Reynolds number aerodynamics
- Design of novel drone types Design of novel MAV types Design of novel MAV types (hybrids, propulsion, silent drones)
- MAVs that interact robustly with their environment
- New control methods
- Reconfiguration of MAVs in unpredicted events
- Autonomous navigation in GPS-denied environments
- Integration of MAVs in airspace Integration of MAVs in airspace (sense-and-avoid, UTM, . . .)
- Application of MAVs in industry, agriculture, environmental monitoring, conservation, search and rescue operations

Venue

- Location: Bristol, United Kingdom
- Event dates: September 16 – 20, 2024. Web site: http://www.imavs.org/2024

http://www.imavs.org/

Conference Oral Presentations

Session 1: Bioinspiration and Flight Operations I

Enhancing Perching Capabilities of Flapping Wing Robots with Silicone-based Electromagnetic Claws p.15
Flight control of an insect-like tailless flapping wing nano-robot around hover p.23
A Bio-Inspired Micro Air Vehicle and the Experimental Setup for Aerodynamic Testing p.33
Bumblebees Exhibit Adaptive Flapping Responses to Air Disturbances p.41
Regulatory Compliant UAS Navigation Based on a Bitmap Grid Local Path Planning Algorithm p.49

Session 2: Machine Learning

Robust Heading Estimation from Polarization Images by Deep Neural Networks p.56
Pose Estimation with a Lightweight Visual-Inertial Neural Network for Agile UAV Flights p.64
Evolving Spiking Neural Networks to Mimic PID Control for Autonomous Blimps p.72
Accelerating Swarms : Harnessing Hardware Acceleration and Parallelization in Multi-Agent... . . . . p.79
Adaptive Neural Network Quadrotor Trajectory Tracking Controller Tolerant to Propeller Damage p.87
Binary Networks and Continual Learning for Pose Estimation from a Single Aerial Image p.100

Session 3: Platform Design

Feasibility Study of a Solar Powered Hybrid Micro-Aerial Vehicle p.109
A nonlinear autopilot for a piloted Quad-plane p.116
Spherical Aerial Manipulator Robot for Exploration in Complex Forest Environments p.124
Dexterity Assessment of Fully-Actuated UAVs p.131
Design of a Tapered-Wing Tiltrotor UAV for Multi-Mission Applications p.139
Effect of Aerodynamic Summation on Multirotor UAV Hover Performance p.148

Session 4: Sensing

Swarm based airborne wind measurement p.156
Landing Zone Detection for MAVs using Depth Images and Vision Transformers p.163
Crossing Gate Detection Using Audio Frequency Pitch Tracking p.171
Comparative Analysis of Cameras for ArUco Marker Recognition in Unmanned Aerial Vehicles p.177
Photovoltaic plant monitoring and inspection through synergic integration of UAVs and IoT p.185
Flying a Quadrotor with Unknown Actuators and Sensor Configuration p.193

Session 5: Bioinspiration and Flight Operations II

Novel Flapping Wing Micro Aerial Vehicle Design Featuring a Bio-inspired Abdomen Mechanism p.200
A method for determining the safe separation to an unknown-model uncooperative UAV in U-space p.209
A Detect And Avoid concept for UAS operations near Hospital Emergency Medical Services helipads in... . . . . p.219
Extended Reception Coverage and Spoofing Localization of Direct Remote ID utilizing Ad-Hoc Mesh... . . . . p.228
Effects of Simultaneous Turbulence and Ice Accretion on Small Fixed-Wing UAV Performance p.236
Gen4jectory algorithm – 4-D trajectory planning with minimised flight time for multiple rotary-wing... . . . . p.243

Session 6: Control

Control of the DarkO Tail-Sitter Drone through an LMI-Based Static Output Feedback Design p.254
A Study of Gain Tuning for Prolonged Physical Contact of Fully-Actuated UAVs p.262
Restricted Model Control applied to the Quanser Aero with uncertain parameters p.270
On the Lateral Non-Stabilizability of PID Controllers for Vision-Based Line-Following Multicopters... . . . . p.277
Swarm-in-Blocks: Simplifying Drone Swarm Programming with Block-Based Language p.285
Comparative analysis of ESC and Flight Controller Communication Protocols and their impact on... . . . . p.293

Session 7: Aircraft Performance and Conservation

Exploring the potential of collaborative UAV 3D mapping in Kenyan savanna for wildlife research p.301
Effects of Wind Speed on Quadcopter Ceiling Effect p.309

http://www.imavs.org/

Drone Swarms for Animal Monitoring: A Method for Collecting High-Quality Multi-Perspective Data p.316
WildBridge: Conservation Software for Animal Localisation using Commercial Drones p.324

http://www.imavs.org/

Keynote Speakers

WildDrone

Professor Ulrik Pagh Schultz Lundquist, University of Southern Denmark

Abstract

The WildDrone project aims to revolutionize wildlife conservation practices by using autonomous drone technology as a unifying platform to monitor wildlife populations, track their movements, and manage human-wildlife conflicts. In this talk, Professor Pagh Schultz Lundquist will give an overview of the WildDrone network, research and training activities, the interdisciplinary methodology, and the specific drone-related research currently taking place in the project.



Biography

Professor Ulrik Pagh Schultz Lundquist is Professor and Head of Center for SDU UAS, the drone center at University of Southern Denmark. He is currently coordinator of the WildDrone Marie Curie Network and is interested in the design and implementation of aerial, mobile, and self-organizing robotic systems with a focus on the use of high-level software abstractions to achieve safety, reliability, and robustness.

Enabling UAVs to use tools in complex dynamic environments

Richard Green – University of Canterbury

Abstract

Close-proximity operation of UAVs using tools in outdoor environments requires a level of control precision (sub centimetre) and position estimation robustness/accuracy beyond what is currently possible. We will discuss our approaches to solve this by improving multirotor UAV position control precision, especially in windy conditions – and by significantly increasing the robustness/accuracy of visual-inertial position/orientation estimation in difficult environments. Our progress is evidenced by a fully autonomous UAV using tools to prune pine trees, prune branches near power lines, harvest catkins, test power-line joints – and working towards using tools (e.g. screwdriver) in construction applications.



Biography

Richard Green has been lecturing in computer science at the University of Canterbury since 2014 after running his own successful software startup in Sydney. With over 200 refereed publications, Richard heads the Computer Vision Research Lab with an emphasis on autonomous robot vision/deep learning (robots/UAVs/underwater robots). Richard has led numerous successful MBIE-funded projects, including an autonomous robot-vine-pruner project and an autonomous forest-pruning-drone project, and currently leads a $10m MBIE project for drones using precision tools along with a $6m MBIE project automating vineyard robots.

http://www.imavs.org/

MIMRee: Multi-Robot Inspection and Maintenance of Offshore Wind Turbine Blades

Simon Watson – University of Manchester

Abstract

The inspection and maintenance of offshore wind turbine blades is critical to long-term operations, however their location makes human interventions hazardous and costly. UAVs have been used for remote visual inspections, however interventions are much more difficult. This presentation will showcase work on the remote deployment of ground-based robots for intervention missions by UAVs.



Biography

Dr Simon Watson is a Reader in Robotic Systems at the University of Manchester. His research is focused on the development of robotic platforms for the inspection of critical infrastructure assets in the net zero energy and urban infrastructure domains. He has developed aerial, aquatic and ground platforms for the nuclear, offshore wind and transportation sectors and was the co-founder of the spin-out company Ice9.

Bio-informed Guidance and Control: From Birds to Morphing Air Vehicles

Graham Taylor – University of Oxford

Abstract

Birds achieve manoeuvrability far exceeding that of fixed-wing vehicles by morphing their wings and tail to vary flight performance and exert fine control. Presenting data from many birds and thousands of flights recorded in a state-of-the-art motion capture lab, this talk will explore how birds optimize and control visually-guided behaviours like perching and collision avoidance. The problems that birds have adapted to solve are the same as faced by aerial robots in natural and built environments, so insights from birds have much to offer the design of uncrewed air systems – especially in the era of environmental robotics.



Biography

Professor Graham Taylor is Deputy Head of the Mathematical, Physical and Life Sciences Division at the University of Oxford, where he leads the Oxford Flight Group. His research focusses on the guidance, navigation and control of animal flight and its applications to autonomous systems, spanning lab, field and computational studies.

http://www.imavs.org/

Papers

[1] Enhancing Perching Capabilities of Flapping Wing Robots with Silicone-based Electromagnetic Claws . . . p.15
[2] Flight control of an insect-like tailless flapping wing nano-robot around hover . . . p.23
[3] A Bio-Inspired Micro Air Vehicle and the Experimental Setup for Aerodynamic Testing . . . p.33
[4] Bumblebees Exhibit Adaptive Flapping Responses to Air Disturbances . . . p.41
[5] Regulatory Compliant UAS Navigation Based on a Bitmap Grid Local Path Planning Algorithm . . . p.49
[6] Robust Heading Estimation from Polarization Images by Deep Neural Networks . . . p.56
[7] Pose Estimation with a Lightweight Visual-Inertial Neural Network for Agile UAV Flights . . . p.64
[8] Evolving Spiking Neural Networks to Mimic PID Control for Autonomous Blimps . . . p.72
[9] Accelerating Swarms : Harnessing Hardware Acceleration and Parallelization in Multi-Agent... . . . p.79
[10] Adaptive Neural Network Quadrotor Trajectory Tracking Controller Tolerant to Propeller Damage . . . p.87
[11] Binary Networks and Continual Learning for Pose Estimation from a Single Aerial Image . . . p.100
[12] Feasibility Study of a Solar Powered Hybrid Micro-Aerial Vehicle . . . p.109
[13] A nonlinear autopilot for a piloted Quad-plane . . . p.116
[14] Spherical Aerial Manipulator Robot for Exploration in Complex Forest Environments . . . p.124
[15] Dexterity Assessment of Fully-Actuated UAVs . . . p.131
[16] Design of a Tapered-Wing Tiltrotor UAV for Multi-Mission Applications . . . p.139
[17] Effect of Aerodynamic Summation on Multirotor UAV Hover Performance . . . p.148
[18] Swarm based airborne wind measurement . . . p.156
[19] Landing Zone Detection for MAVs using Depth Images and Vision Transformers . . . p.163
[20] Crossing Gate Detection Using Audio Frequency Pitch Tracking . . . p.171
[21] Comparative Analysis of Cameras for ArUco Marker Recognition in Unmanned Aerial Vehicles . . . p.177
[22] Photovoltaic plant monitoring and inspection through synergic integration of UAVs and IoT . . . p.185
[23] Flying a Quadrotor with Unknown Actuators and Sensor Configuration . . . p.193
[24] Novel Flapping Wing Micro Aerial Vehicle Design Featuring a Bio-inspired Abdomen Mechanism . . . p.200
[25] A method for determining the safe separation to an unknown-model uncooperative UAV in U-space . . . p.209
[26] A Detect And Avoid concept for UAS operations near Hospital Emergency Medical Services helipads in... . . . p.219
[27] Extended Reception Coverage and Spoofing Localization of Direct Remote ID utilizing Ad-Hoc Mesh... . . . p.228
[28] Effects of Simultaneous Turbulence and Ice Accretion on Small Fixed-Wing UAV Performance . . . p.236
[29] Gen4jectory algorithm – 4-D trajectory planning with minimised flight time for multiple rotary-wing... . . . p.243
[30] Control of the DarkO Tail-Sitter Drone through an LMI-Based Static Output Feedback Design . . . p.254
[31] A Study of Gain Tuning for Prolonged Physical Contact of Fully-Actuated UAVs . . . p.262
[32] Restricted Model Control applied to the Quanser Aero with uncertain parameters . . . p.270
[33] On the Lateral Non-Stabilizability of PID Controllers for Vision-Based Line-Following Multicopters... . . . p.277
[34] Swarm-in-Blocks: Simplifying Drone Swarm Programming with Block-Based Language . . . p.285
[35] Comparative analysis of ESC and Flight Controller Communication Protocols and their impact on... . . . p.293
[36] Exploring the potential of collaborative UAV 3D mapping in Kenyan savanna for wildlife research . . . p.301
[37] Effects of Wind Speed on Quadcopter Ceiling Effect . . . p.309
[38] Drone Swarms for Animal Monitoring: A Method for Collecting High-Quality Multi-Perspective Data . . . p.316
[39] WildBridge: Conservation Software for Animal Localisation using Commercial Drones . . . p.324

http://www.imavs.org/

Authors

A

Afonso, Savio Pessôa p.177
Aguado, Mauro Villanueva p.87
Alaez, Daniel p.49
Almeida, Agnes p.285
Andersen, Klavs p.219
Arias-Perez, Pedro p.185
Armanini, Sophie p.15
Azorin-Molina, Cesar p.236

B

Bannwarth, Jérémie p.148
Becker, Marcelo p.293
Blaha, Till M. p.193
Boumediene, Mouad p.243
Bronz, Murat p.79
Buchanan, Rory p.148
Burgers, Tim p.72
Buzzatto, Joao p.131

C

Cabrera-Ponce, Aldrich A. p.171
Campoy, Pascual p.185
Carlier, Nicholas p.236
Carrierou, Amelie p.270
Chan, Katrina p.262
Chan, Woei-Leong p.33
Chanchu, Lohithsai Yadala p.56
Chen, Hung-Yu p.33
Chin, Chia-Le p.33
Christensen, Anders p.316
Condomines, Jean-Philippe p.270
Cotrim, Reinaldo Monteiro p.177
Croon, Guido de p.56, p.72, p.87
Cruz-Vega, Israel p.124
Curpanaru, Armand-Ioan p.254

D

Demir, Muhammet p.15
Demourant, Fabrice p.254
Duran, Kevin Guzman p.64

E

Elliott-Roe, Rhiannon p.309
Estrada, Sergio Araujo p.139

F

Fernandez-Cortizas, Miguel p.185
Fisher, Alex p.41
Floren, Merijn p.23

G

Gan, Wentian p.301
Gavin, Timothée p.79
Giles, Alejandro p.64
Gp-Lenza, Guillermo p.185
Grøntved, Kasper p.316
Gutierrez-Giles, Alejandro p.124

H

Hammad, Ahmad p.15
Helsen, Bart p.23
Herlev, Victor Dahl p.228
Hernandes, André Carmona p.293
Hine, Duncan p.324
Holm, Pieter Van p.277
Huang, Chia-Hsiang p.33
Høffer, Mads Fogh p.228

J

Jakobi, Timothy p.41
Jensen, Kjeld p.209, p.219, p.228, p.243
Jepsen, Jes p.209, p.219, p.228
Johnson, Caspian p.324
Join, Cedric p.270
Junio, Durval Ferreira Sobrinho p.177

K

Kay, Nicholas p.148, p.236
Kazemi, Shahab p.131, p.262
Kean, Sebastian p.156
Kirk, Jonty p.262
Kline, Jenna p.316

L

Lacroix, Simon p.79
Laporte-Devylder, Lucie p.316
Leng, Jiaxun p.200
Lima, Fábio Santos p.177
Lundquist, Ulrik p.316

http://www.imavs.org/

Lustosa, Leandro Ribeiro p.277

M

Ma, Trista Yuting p.236
Maalouf, Guy p.228, p.324
Mallikarjuna, Apoorva Busunur p.277
Marino, Matthew p.156
Marsicano, João Aires p.285
Martinez-Carranza, José . . p.64, p.100, p.124, p.163, p.171
Martínez-Ramírez, Marco Antonio p.116
Martínez-Velasco, Diego Alexis p.116
Meier, Kilian p.324
Melero-Deza, Javier p.185
Midtiby, Henrik Skov p.243
Mohamed, Abdulghani p.156
Morelli, Luca p.301

N

Ng, Steven p.200

O

Olivares-Figueroa, Daniel p.124
Oliveira, Rafael Gomes de p.177
Ortiz, Manuel Isidro Martin p.100
Ozer, Özgün p.309

P

Panov, Ivan p.209, p.228, p.243
Pascau, Pablo De Porcellinis p.49
Perez-Saura, David p.185
Perez-Segui, Rafael p.185
Pirooz, Amir p.236
Ponce, Aldrich Alfredo Cabrera p.100
Presser, Mirko p.209
Prieto, Manuel p.49
Pérez, Carlos Santos p.185
Pérez, Leticia Oyuki Rojas p.100

R

Rascon, Caleb p.171
Ravi, Sridhar p.41
Remes, Bart D.W. p.193
Remondino, Fabio p.301
Ribeiro, Vitor Garcia p.293
Richards, Arthur p.324
Richards, Peter p.148
Richardson, Tom p.324
Rodríguez, Edgar Ulises Rojo p.139
Rodríguez, Erik Gilberto Rojo p.139
Rodríguez-Cortés, Hugo p.116
Roelandt, Thomas p.23
Rojas-Perez, L. Oyuki p.171
Rolland, Edouard p.316

S

Saeed, Burhan p.109
Salazar, Octavio García p.139
Sansou, Florian p.254
Santos, Leandro Goncalves dos p.177
Shukla, Vandita p.301
Singh, Chantelle p.131, p.262
Smeur, Ewoud J.J. p.193
Soria, Carmelo Simon p.277
Souza, Jeovana Miranda p.177
Souza, João Vitor Nascimento p.177
Stam, Daphne p.56
Stol, Karl p.131, p.148, p.262
Stroobants, Stein p.72
Sánchez, Francisco Javier Rodríguez p.185

T

Tang, Shun p.200
Thomas, Peter p.109
Tradacete-Agreda, Miguel p.185
Trybala, Pawel p.301

V

Vandepitte, Dirk p.23
Vazquez, Victoria p.163
Vazquez-Meza, Victoria Eugenia p.171
Villadangos, Jesus p.49
Villela, Guilherme Barros p.293
Visser, Coen C. de p.193

W

Wagter, Christophe De p.87
Wang, Xin p.301
Watkins, Simon p.41, p.156
Watson, Matthew p.324
Winkler, Erik p.228
Wood, Kieran p.309

Y

Yu, Yifei p.301

Z

Zuidgeest, Ruben p.56

http://www.imavs.org/

http://www.imavs.org/

Release

Released on January 20th 2025.

V1.2, Rev.16822, 2026-10-18 09:36
Created by: T. Richardson, B. Kocer and C. De Wagter

Enhancing Perching Capabilities of Flapping Wing Robots with Silicone-based Electromagnetic Claws

Ahmad Hammad[1*], Muhammet Demir[1], and Sophie F. Armanini[1,2]
Chair of eAviation, Department of Aerospace and Geodesy, TUM School of Engineering and Design, Technical University Munich, Germany[1]
Department of Aeronautics, Imperial College London, UK[2]

http://www.imavs.org/

ABSTRACT

Flapping wing micro air vehicles (FWMAVs) have garnered significant attention over the past few decades for their potential in various applications. However, developing an effective grasping mechanism for these vehicles still poses a critical challenge. Researchers have explored several solutions focusing on FWMAV perching, which are mainly focused on either cylindrical or planar perching. This study presents a novel approach to FWMAV perching based on a magnet-based claw structure utilizing an electromagnet integrated within a silicone-based soft gripper. The proposed solution enables perching on circular, cylindrical, and planar surfaces. The goal was to optimize the claw's ability to maintain stable perching across different perch geometries. The magnetic claw was tested under various conditions and on multiple surfaces to determine the feasibility of the perching mechanism. Additionally, the mechanism was integrated on two different commercially available platforms for further testing, and they demonstrated promising results. Results demonstrate the current approach's effectiveness and give insight into the limitations. For example, a major limitation is the ability to only perch on ferromagnetic surfaces. In conclusion, by providing comprehensive insight into the design, development, manufacturing, and testing of a magnetic claw-based solution, this research advances our fundamental understanding of the challenges faced by FWMAV perching mechanisms and proposes a solution. This work opens new avenues for the real-world use of FWMAVs by addressing the critical problem of grabbing and perching.

1 INTRODUCTION

The demand for micro air vehicles is continuously increasing. Their applications include surveillance, data collection, outdoor sensing, traffic control, mapping of unknown areas, search and rescue, disaster response, and delivery. The mission capabilities of micro air vehicles are expanding with today's technological advances. However, several challenges remain, which include a silent operation during surveillance missions and operating safely when close to nature, i.e., humans or trees/flowers, etc. Flapping wing micro aerial vehicles (FWMAVs) use a flapping wing motion to fly and produce less noise than the rotor blades of conventional micro air vehicles. They are also generally safer to operate thanks to their low weight, flexible wings, and lack of propellers. Therefore, the popularity of FWMAVs is gradually increasing. These bioinspired micro aerial vehicles mimic the motion of natural flyers, like birds and insects, generally operate in a low Reynolds number regime, and have highly complex aerodynamics.

However, micro-aerial vehicles (MAVs), and especially FWMAVs, have drastically limited endurance. This can be addressed by using a grasping mechanism that will assist the vehicle to attach or grasp on a surface and, thus, consume little to no energy while remaining perched. This will increase the overall endurance. This will significantly improve the applications of FWMAVs, which are generally safer and silent; however, they have limited payload capacity and thus have limited endurance.

Researchers have proposed various perching mechanisms in the past, e.g., as discussed by Meng *et al.*[1] and Hammad *et al.*[2] in their respective reviews. Some of the examples are discussed here to give a perspective of existing advancements in the field of aerial grasping. Grasping has been studied extensively for multi-rotor aerial vehicles [3, 4, 5, 6, 7]. A bioinspired, underactuated perching mechanism weighing 478g was developed by [8]. A 9g claw-like grasping mechanism was installed and tested on a 25g vehicle by [9]. A passively actuated, lightweight mechanism weighing 28g mounted on a 593g vehicle was developed by [10]. Broers and Armanini [11] developed a lightweight, passively actuated gripper mechanism inspired by the fin ray effect.

Attaching is also another mechanism used for effective perching. Rather than using the conventional grasping method to perch, these vehicles use magnets [12], adhesives [13], or vacuum pumps [14] to attach themselves to the surface. For very small-scale FWMAVs, magnets have been used as a mode for perching [15]. Most of the work with such mechanisms has been done for conventional non-

*Corresponding author: ahmad.hammad@tum.de

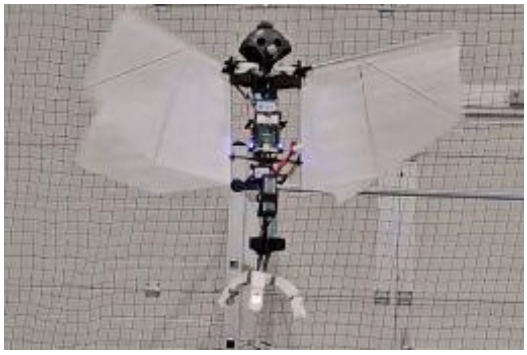


Figure 1: Magnetic Gripper Installed on Entomopter

flapping wing vehicles. A micro-spine-based attaching mechanism for a fixed-wing UAV enabling it to perch vertically on the surface was developed [16, 17, 18]. The use of micro-spines as a means for attaching mechanisms for MAVs was also demonstrated by [19, 20].For MAVs perching, dry adhesives [21] and electro-adhesives [22] have also been used. A heavier quad-rotor weighing 1.77kg was equipped with a vacuum pump to enable perching [23].

Most existing research on grasping and perching has focused on quadrotors and fixed-wing vehicles. Only limited work has been done to investigate perching mechanisms for FWMAVs. Gomez *et al.* [24] developed an SMA-actuated perching mechanism. This mechanism relied on active actuation. An ultra-lightweight mechanism was installed on an ornithopter by [12] and used magnets to hold onto the surface. Another mechanism with carbon fiber claws, passively closed by a spring and used a servo motor to open the claws again, was developed by [25]. Attaching-based perching mechanisms are also quite popular, and recently, Lau *et al.* [26] studied electro-adhesive and dielectric elastomers-based perching mechanisms for FWMAVs.

The above research shows that advancement in perching mechanisms for FWMAVs is still required, especially since there is a need for mechanisms that can adapt to various surfaces. It also highlights the need for a perching mechanism to not only improve the efficiency of a vehicle but also make it more versatile, hence making it more useful for different types of missions and essentially making it easier to achieve autonomous operation in the long term. This gap is covered with the presented mechanism. This paper presents a magnetic-based, lightweight perching mechanism that can adapt to various surfaces and increase the overall flight time of FWMAVs (Figure. 1). The designed mechanism demonstrates successful perching both on cylindrical and planar ferromagnetic surfaces. The mechanism combines grasping and attaching techniques and utilizes an electromagnet in a lightweight silicone-based claw adaptive to circular, cylindrical, and spherical surfaces. The electromagnets attach themselves to the required surface, while the silicone-based soft gripper is used to grasp the object. The mechanism is easier to control and requires a simple actuation. Details of the mechanism are given further in section 2.

2 SYSTEM OVERVIEW

The requirements mentioned in section 1 demand a novel approach and combine grasping and attaching mechanisms to develop an effective perching system. Various grasping methods were considered, and a silicone-based robotic gripper mechanism was finalized. Soft robotics has shown promising progress recently, and their applications are becoming wider with further research [27, 28]. The silicone-based gripper was coupled with an electromagnet, thus providing more active control. Secondly, the drone did not show adequate thrust to release the gripper once takeoff was required; hence, the choice was made to use an electromagnet rather than a permanent magnet. The gripper mechanism was attached to a commercially available Entomopter weighing 114g and an Ornithopter weighing 450g. This section covers the magnets, silicone gripper design, and vehicle characteristics used in the current study.

2.1 Electromagnets

Electromagnets were selected for the perching mechanism as they provided more control. Moreover, using magnets provided the advantage of attaching to various surface shapes, e.g., circular, cylindrical, and planar. They are easier to actuate, thus eliminating the use of more complex actuation mechanisms and simplifying the transition from perched to flying mode.

It is crucial to select an appropriate electromagnet since a stronger magnet would also become heavier and require a bigger battery. The vehicle may drop while perched if the magnetic force is too low. The respective spacing between the magnets is also an important parameter to consider since the magnets were required to be installed and help the silicone-based claw to close, especially for circular and cylindrical surfaces. Keeping the space too small would mean excessive use of magnets, which was also unfavorable. Available data suggested that the distance should be more than 10mm to have effective perching and to avoid attractive and repulsive forces between the magnets [29]. The holding force is also essential, especially in selecting the appropriate electromagnet and the battery used for actuation. For the current design, it was desired that the manufactured gripper be widely applicable rather than optimized the design for a specific FWMAV. Therefore, it was decided that the claws could hold FWMAVs weighing as low as 100g to as much as 800g. Electromagnets of strengths 2N and 3N were tested with different batteries to determine the effect of the battery on the holding force. Table. 1 details the tested batteries and the output. It was decided to use the 3s battery for experimentation with the 2N Electromagnet. The measured holding force was also deemed reasonable for perching tasks.

http://www.imavs.org/

Battery/Voltage	3N Electromagnet	2N Electromagnet
3.7V(1S)	1.3N	1.49N
7.4V(2S)	2.39N	3.05N
11.1V(3S)	3.74N	3.85N
12V	4.64	4.03N
14.8V(4S)	5.3N	4.52N

Table 1: Physical and Kinematic data of Ornithopter [30]

Four magnets of strength 2N each were used and housed inside the silicone-based claw. Such a configuration provides enough gripping strength to land a vehicle weighing 0.2kg to 0.8kg. Additional weight may also be carried, and only a change in the electromagnet is required, whereas the complete design may remain the same. For bigger vehicles, using a two-claw mechanism is also a suitable option.

2.2 Gripper Design

The claw was constructed from silicone since it provides adequate flexibility, is easier to manufacture, is lightweight, durable, and easily molded. A three-finger design in Y configuration was selected for simplicity. The length of each finger was kept at 60mm to ensure that the gripper can grasp on a range of surface sizes, especially for cylindrical perching, and that it does not become too heavy. The magnets alone can also successfully demonstrate perch on different ferromagnetic objects [15, 12]. This, however, becomes difficult, especially for larger vehicles that do not possess vertical takeoff and landing capability. Perching at a certain angle can still be managed such that the vehicle's center of gravity is always in line with the magnets, and no additional moment is generated due to angular perching. This can be overcome by using the claws since, for a wider range of angles now, the center of gravity remains within the claw range, ensuring a successful perch. This additional advantage led to using silicone-based claws as part of the gripper mechanism.

Electromagnets were placed at the tips of the claw and one at the center to provide adequate support during perching. The mold was prepared by 3D printing, and silicone was poured inside. It is pertinent to mention that the mixing ratio is essential; otherwise, it may result in a too-hard claw that does not bend or becomes too soft and bends with its weight. Leaving air while pouring also causes brittleness to the claw. The electromagnets were housed inside the holes seen in Figure. 2. Wire channels were made within the design so the wires powering the electromagnets do not create hindrances during perching. The final weight of the gripper was 42g. Since neither of the two FWMAVs used to test the mechanism were intended to carry any payload, lightweight is essential.

A spacer was designed to attach the claw to the respective vehicle (Figure. 3). The spacer made it easier to mount the claw on the FWMAV, but experiments showed that it could

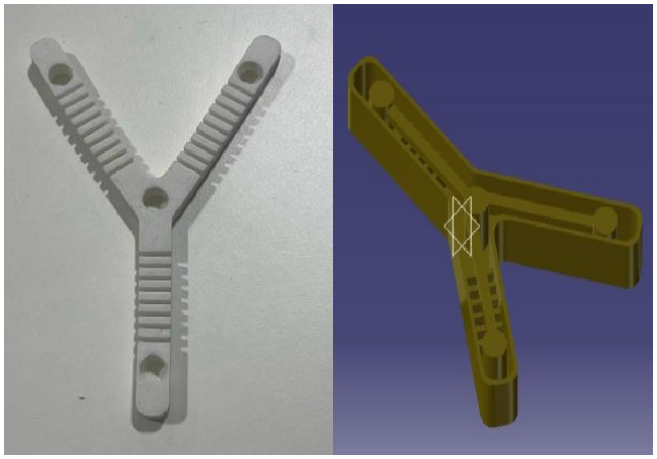


Figure 2: Claw Structure (left) and CAD Model (right)

fall off due to high vibrations during flight. Therefore, an adhesive was used between the spacer and the claw to avoid such failure during flight. Actuation of the electromagnets was a simple process. A transistor was required, connected to the flight controller, which was used to actuate the magnets. A 900 Ohm resistor was used to limit the current passing through; otherwise, it could lead to transistor damage. Various configurations were tested to place the transistor in the circuit.

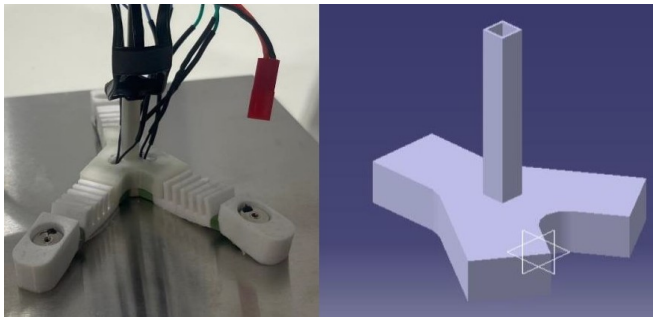


Figure 3: silicone-based claw (left) and spacer (right)

2.3 Drone Platforms

Since the goal was to develop a generic system that may be applied to multiple vehicles, it was decided to demonstrate its effectiveness and functionality on two significantly different platforms. These platforms are further detailed in this section and shown in Figure. 4.

2.3.1 Test Platform 1

Table. 2 gives the detailed parameters of the Entomopter developed by Flapper Drones [31] used for the study. The vehicle is based on the DelFly Nimble [32]. This FMWAV was chosen because it shows vertical takeoff and landing ability,

http://www.imavs.org/

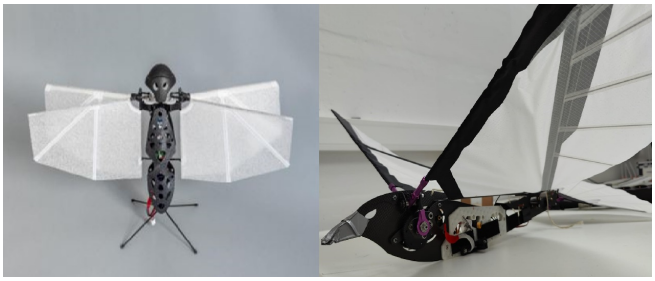


Figure 4: Entomopter (Left) and Ornithopter (Right)

making it easier to test the developed mechanism. The critical challenge to address for this vehicle was the weight constraint since, for the mechanism weight, we had to remain within the allowable range.

Parameters	Values
Wingspan	0.049m
Weight	102 g (min. take-off weight) 127 g (max. take-off weight)
Payload Capacity	60g (shells & landing gear removed)
Flight Time	8 min (forward 3 m/s, min. weight) 5 min (hover, max. payload)

Table 2: Characteristics of the Entomopter [31]

The total weight of the Entomopter includes electronics and the structure. The flapper drone has a recommended payload capacity of 25g [31] with all components on board. For the current case, the protective shells, the existing landing platform, and other additional components that are not required were removed, increasing the vehicle's payload capacity. This payload capacity was tested in the lab, and it was observed that its effect on the overall endurance of the vehicle was not that significant. A relatively lightweight gripper weighing 32g with a 40mm finger length was also tested during the design process. However, a smaller gripper size leads to an unstable perch for the second test platform due to its larger surface area, lack of VTOL capability, and higher moment while perching. This was also confirmed with experimentation. Therefore, further experimentation was performed with the design suitable for both vehicles, which is presented in the paper.

2.3.2 Test platform 2

The Orthinopter platform from Ornihobby [33] was used to test the perching mechanism. Characteristics of the vehicle are given in Table. 3.

Parameters	Values
Mean chord length of each wing	0.455 m
Aspect ratio	1.42
Mass of the ornithopter	450 g
Flight speed of the ornithopter	10–25 km/h
Range of wingbeat frequency	3.5–4.5 Hz
Half Wing Span	70 cm

Table 3: Physical and Kinematic data of Ornithopter [30]

3 RESULTS AND DISCUSSION

The claw mechanism was tested on different ferromagnetic surfaces to ensure it could withstand the loads it was subjected to and successfully land on objects of various sizes and shapes. Experiments were conducted with the mechanism alone and the mechanism attached to the FWMAV test platforms. The obtained results are shown in this section.

3.1 Static Tests

Static tests were conducted to check the strength and adaptability of the claw, first on its own and then while attached to an FWMAV vehicle. In particular, the goal was to evaluate whether the mechanism could attach successfully to cylindrical and planar surfaces and whether the electromagnets supplied the required forces.

The claw in its open state can be seen in Figure. 5 (left). Figure. 5 (right) shows the claw in its closed state attached to a cylindrical rod, while Figure. 6 shows the flaw attached to a ferromagnetic planar surface. In these tests, the claw was manually attached to the different structures, and the electromagnets were activated and able to attach successfully to the metallic surfaces.

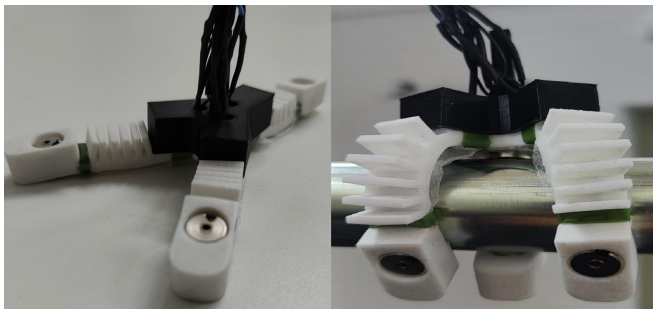


Figure 5: Claw in Open (left) and closed (right) state

These tests demonstrate the adaptability of the claw to different structure shapes. After proving that the basic design requirements were met, the claw was installed on the FWMAV for further testing.

Figure. 7 shows static tests performed with the claw attached to the FWMAV. Successful perching was demonstrated for both the planar and the cylindrical surfaces. An inverted perching test was also conducted where the FWMAV was hung upside down on the cylindrical surface and then attached to the surface by activating the magnets, which suggests that the claw mechanism is strong enough to withstand

http://www.imavs.org/



Figure 6: Magnetic-claw attached to Planar Surface

the complete weight of the Entomopter.

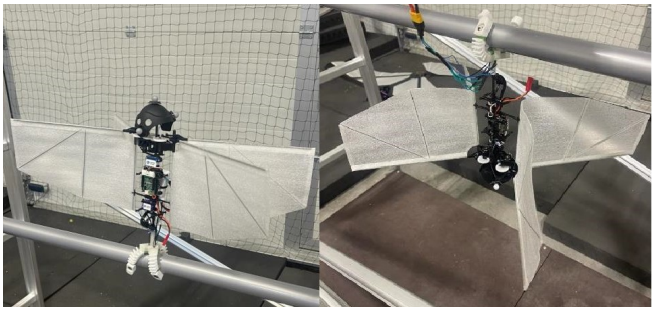


Figure 7: Cylindrical and Inverted Perching

Static tests were also conducted to demonstrate successful attachment while installed on the Ornithopter. A rectangular bar with rounded edges was used to show the adaptability to the different shapes and that the silicone-based electromagnetic claw mechanism can withstand the vehicle's weight and perform successful perching tasks. This is shown in Figure. 8.

3.2 Flight Experiments

Flight experiments were conducted on the Entomopter to evaluate the feasibility of the designed claws in flight and perching tasks. The claws demonstrated successful perching on various surfaces. Cylinders of various sizes were used to analyze the perching capabilities of the claw mechanism. The smallest radius on which the claw demonstrated a successful perch was 17mm Figure. 9 (left). Landing on circular cylinders with radii greater than that was also demonstrated. Since the surface tends to become more flat as the radius increases, landing on cylinders with larger diameters was relatively straightforward, as seen in Figure. 9 (right). Since the gripper mechanism is actuated actively, the electromagnets

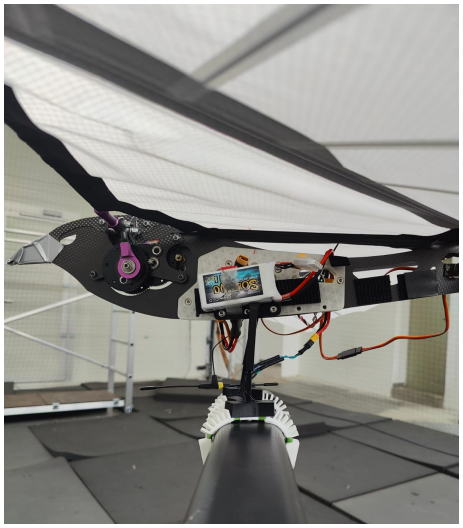


Figure 8: Static Test on Ornithopter

are already turned on when contact is made. Thus, the attachment of these electromagnets with the ferromagnetic surface is instant.



Figure 9: Landing on Cylinders of Different Radii

Similarly, landing on a planar surface was performed and is shown in Figure. 10. This showed the overall versatility of the claw mechanism and its adaptability to various shapes. While perching on the cylinders of smaller radii, it was observed that a greater approach velocity is required than landing on cylinders with larger diameters. This was owing to the increased bending requirement for the silicone-based structure to successfully envelop the cylinder. A higher impact force ensured that the fingers would wrap around the cylinder and the subsequent engagement of magnets with the surface.

The electromagnets were integrated with the onboard flight controller and connected to a separate channel on the transmitter. The activation was done by simply switching the channel on. This made the whole actuation process simpler. The silicone-based claw's bending occurs under the vehicle's weight and the impact force with the surface. There was no

http://www.imavs.org/

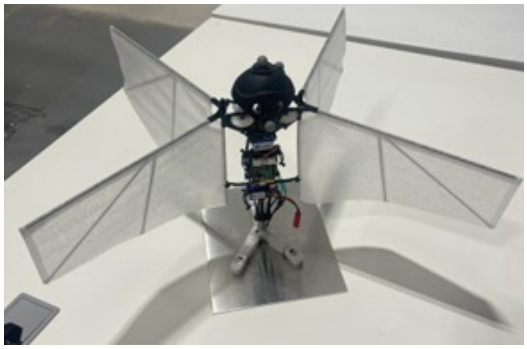


Figure 10: Planar Landing

additional actuation requirement for the bending as it occurs passively.

Another foreseen issue in the claw mechanism was planar perching. The claws were to open when hitting a planar surface and not curl up, which would lead to a failed perch. This issue was addressed at the design level while the silicone structure was prepared. The mixing ratio was maintained at an appropriate level so that the silicone-based structure had enough stiffness that the claw did not curl up upon impact. It was also noted that too high of stiffness can result in a failed perch during cylindrical perching as the claw will not close.

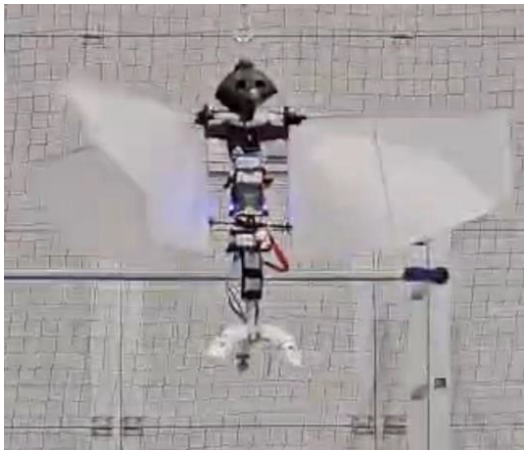


Figure 11: Flight after Perching

Figure. 11 shows successful takeoff and subsequent flight from a perched position. The detachment mechanism was simple, as the electromagnets could switch off swiftly. It should be noted that with the electromagnets turned off, the FWMAV could stay in a perched position for some time. However, while taking off from an inclined perched position, it will instantly become unstable and start to fall. Therefore, the wing flapping sequence should begin once detachment is initiated.

The above experiments demonstrate successful perching on planar and cylindrical ferromagnetic surfaces. Static tests for both Entomopter and Ornithopter show a wide range of capabilities. Using electromagnets is promising, and housing them inside a silicone-based claw gives more adaptability to different surfaces. The tests further show that the mechanism can be applied to vehicles of various sizes and configurations and can hold a variety of vehicles with varying weights up to 600g (including the weight of the mechanism and supports). Moreover, the mechanism could be easily adapted to achieve higher holding strength, e.g., by changing the electromagnet used without changing the overall design of the claw.

4 CONCLUSION

With the goal of enabling perching of lightweight FWMAVs on both cylindrical and planar surfaces, a novel electromagnetically actuated silicone-based claw mechanism was designed. A prototype of the proposed concept was manufactured, weighing just 42g. It was tested for suitability on two FWMAVs, an insect-like Entomopter weighing 102g and a bird-like Ornithopter weighing 450g. The design process involved integrating the electromagnets into the silicone-based claw and an in-depth analysis of perching on various ferromagnetic surfaces. The flexible structure of silicone improves its ability to perch on various surfaces. The success achieved in stable perching on various surfaces highlights the potential practical application of this technology in MAVs, particularly in FWMAVs. While significant progress has been achieved, there are still opportunities for further advances and optimization. Future research could focus on optimizing the configuration of the electromagnet, exploring alternative materials, or delving deeper into the dynamics of the perching process. Essentially, this work represents advancing the FWMAV's perching capabilities by developing a magnet-based claw structure. The practical application of this new type of perching mechanism is demonstrated through comprehensive testing, including successful perching experiments on various ferromagnetic surfaces.

REFERENCES

[1] Jiawei Meng, Joao Buzzatto, Yuanchang Liu, and Minas Liarokapis. On aerial robots with grasping and perching capabilities: A comprehensive review. *Frontiers in Robotics and AI*, 8:739173, 2022.

[2] Ahmad Hammad and Sophie Armanini. Landing and take-off capabilities of bioinspiredaerial vehicles: a review. *Bioinspiration & Biomimetics*, 2024.

[3] Long Bai, Hao Wang, Xiaohong Chen, Jia Zheng, Liming Xin, Yupeng Deng, and Yuanxi Sun. Design and experiment of a deformable bird-inspired uav perching mechanism. *Journal of Bionic Engineering*, 18(6):1304–1316, 2021.

[4] Pu Xie and Ou Ma. Grasping analysis of a bio-inspired uav/mav perching mechanism. In *ASME International*

http://www.imavs.org/

Mechanical Engineering Congress and Exposition, volume 56178, page V001T01A012. American Society of Mechanical Engineers, 2013.

[5] Pu Xie, Ou Ma, Zhen Zhao, and Lin Zhang. A bio-inspired uav leg-foot mechanism for landing, grasping and perching tasks. In *AIAA Atmospheric Flight Mechanics Conference*, page 1689, 2015.

[6] William RT Roderick, Mark R Cutkosky, and David Lentink. Bird-inspired dynamic grasping and perching in arboreal environments. *Science Robotics*, 6(61):eabj7562, 2021.

[7] Paul M Nadan, Tatiana M Anthony, Duncan M Michael, Jeffrey B Pflueger, Manik S Sethi, Kelli N Shimazu, Mindy Tieu, and Christopher L Lee. A bird-inspired perching landing gear system. *Journal of Mechanisms and Robotics*, 11(6):061002, 2019.

[8] Courtney E Doyle, Justin J Bird, Taylor A Isom, Jason C Kallman, Daman F Bareiss, David J Dunlop, Raymond J King, Jake J Abbott, and Mark A Minor. An avian-inspired passive mechanism for quadrotor perching. *IEEE/ASME Transactions On Mechatronics*, 18(2):506–517, 2012.

[9] Haijie Zhang, Jiefeng Sun, and Jianguo Zhao. Compliant bistable gripper for aerial perching and grasping. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1248–1253. IEEE, 2019.

[10] HaoTse Hsiao, Jiefeng Sun, Haijie Zhang, and Jianguo Zhao. A mechanically intelligent and passive gripper for aerial perching and grasping. *IEEE/ASME Transactions on Mechatronics*, 27(6):5243–5253, 2022.

[11] Krispin CV Broers and Sophie F Armanini. Design and testing of a bioinspired lightweight perching mechanism for flapping-wing mavs using soft grippers. *IEEE Robotics and Automation Letters*, 7(3):7526–7533, 2022.

[12] Vicente Pérez-Sánchez, Alejandro E Gómez-Tamm, Francisco Javier García-Rubiales, B Arrue, and A Ollero. Analysis of forces involved in the perching maneuver of flapping-wing aerial systems and development of an ultra-lightweight perching system. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1284–1290. IEEE, 2021.

[13] Ludovic Daler, Adam Klaptocz, Adrien Briod, Metin Sitti, and Dario Floreano. A perching mechanism for flying robots using a fibre-based adhesive. In *2013 IEEE International Conference on Robotics and Automation*, pages 4433–4438. IEEE, 2013.

[14] Han W Wopereis, TD Van Der Molen, Tjark Harrie Post, Stefano Stramigioli, and Matteo Fumagalli. Mechanism for perching on smooth surfaces using aerial impacts. In *2016 IEEE international symposium on safety, security, and rescue robotics (SSRR)*, pages 154–159. IEEE, 2016.

[15] Pakpong Chirarattananon, Kevin Y Ma, and Robert J Wood. Fly on the wall. In *5th IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 1001–1008. IEEE, 2014.

[16] Alexis Lussier Desbiens and Mark R Cutkosky. Landing and perching on vertical surfaces with microspines for small unmanned air vehicles. *Journal of Intelligent and Robotic Systems*, 57:313–327, 2010.

[17] Alexis Lussier Desbiens, Alan T Asbeck, and Mark R Cutkosky. Landing, perching and taking off from vertical surfaces. *The International Journal of Robotics Research*, 30(3):355–370, 2011.

[18] Elena Glassman, Alexis Lussier Desbiens, Mark Tobenkin, Mark Cutkosky, and Russ Tedrake. Region of attraction estimation for a perching aircraft: A lyapunov method exploiting barrier certificates. In *2012 IEEE International Conference on Robotics and Automation*, pages 2235–2242. IEEE, 2012.

[19] Morgan T Pope and Mark R Cutkosky. Thrust-assisted perching and climbing for a bioinspired uav. In *Biomimetic and Biohybrid Systems: 5th International Conference, Living Machines 2016, Edinburgh, UK, July 19-22, 2016. Proceedings 5*, pages 288–296. Springer, 2016.

[20] Hai-Nguyen Nguyen, Robert Siddall, Brett Stephens, Alberto Navarro-Rubio, and Mirko Kovač. A passively adaptive microspine grapple for robust, controllable perching. In *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, pages 80–87. IEEE, 2019.

[21] Arash Kalantari, Karan Mahajan, Donald Ruffatto, and Matthew Spenko. Autonomous perching and take-off on vertical walls for a quadrotor micro air vehicle. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4669–4674. IEEE, 2015.

[22] Sanghyeon Park, Daniel S Drew, Sean Follmer, and Juan Rivas-Davila. Lightweight high voltage generator for untethered electroadhesive perching of micro air vehicles. *IEEE Robotics and Automation Letters*, 5(3):4485–4492, 2020.

http://www.imavs.org/

[23] Sensen Liu, Wei Dong, Zhao Ma, and Xinjun Sheng. Adaptive aerial grasping and perching with dual elasticity combined suction cup. *IEEE Robotics and Automation Letters*, 5(3):4766–4773, 2020.

[24] Alejandro Ernesto Gomez-Tamm, V Perez-Sanchez, Begoña C Arrue, and Aníbal Ollero. Sma actuated low-weight bio-inspired claws for grasping and perching using flapping wing aerial systems. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8807–8814. IEEE, 2020.

[25] Raphael Zufferey, Jesus Tormo-Barbero, Daniel Feliu-Talegón, Saeed Rafee Nekoo, José Ángel Acosta, and Anibal Ollero. How ornithopters can perch autonomously on a branch. *Nature Communications*, 13(1):7713, 2022.

[26] Gih-Keong Lau, Chih-Chun Wu, Zhe-Xian Ren, Sudhir Wakler, Shih-Chun Lin, Kuan-Yu Tseng, and Chi-Chung Lu. Lightweight perching mechanisms for flapping-wing drones. In *2023 International Conference on Advanced Robotics and Intelligent Systems (ARIS)*, pages 1–6. IEEE, 2023.

[27] Zhongkui Wang, Damith Suresh Chathuranga, and Shinichi Hirai. 3d printed soft gripper for automatic lunch box packing. In *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 503–508. IEEE, 2016.

[28] Aslan Miriyev, Kenneth Stack, and Hod Lipson. Soft material for soft actuators. *Nature communications*, 8(1):596, 2017.

[29] Stephen Coyle, Carmel Majidi, Philip LeDuc, and K Jimmy Hsia. Bio-inspired soft robotics: Material selection, actuation, and design. *Extreme Mechanics Letters*, 22:51–59, 2018.

[30] Matthew Ng Rongfa, Teppatat Pantuphag, and Sutthiphong Srigrarom. Analysis of kinematics of flapping wing uav using optitrack systems. *Aerospace*, 3(3):23, 2016.

[31] Flapper Drones. Flapper drones - bioinspired flying robots. *https://flapper-drones.com/wp/nimbleplus/*, April 26, 2024.

[32] Matěj Karásek, Florian T Muijres, Christophe De Wagter, Bart DW Remes, and Guido CHE De Croon. A tailless aerial robotic flapper reveals that flies use torque coupling in rapid banked turns. *Science*, 361(6407):1089–1094, 2018.

[33] Ornihobby. Ornihobby - carbonsail ornithipter. *https://www.ornihobby.com/*, April 26, 2024.

http://www.imavs.org/

Flight Control of an Insect-like Tailless Flapping Wing Nano-Robot around Hover

Bart Helsen[1], Thomas Roelandt[1]*, Merijn Floren[1,2] and Dirk Vandepitte[1]
[1]KU Leuven, Celestijnenlaan 300 box 2420, 3001 Leuven, Belgium
[2] Flanders Make@KU Leuven, Leuven, Belgium

http://www.imavs.org/

ABSTRACT

This work describes a control system for a tailless insect-like flapping wing aerial vehicle. The control system is initially implemented in simulation. An experimentally validated rigid wing quasi-steady aerodynamics model represents the vehicle flight dynamics in simulation. A cascaded PID architecture is used to control vehicle flight velocity and attitude. The PID tuning process is automated through a constrained nonlinear optimization process. The attitude estimation process uses onboard accelerometer and gyroscope readings. The accelerometer readings require a low-pass filter to ignore noise from flapping wing related body pitching oscillations. The performance of a complementary filter and extended Kalman filter are considered for the attitude estimation strategy. The flight simulation controller successfully stabilizes the vehicle attitude and follows reference flight maneuvers around hover. The attitude estimation and attitude controller are verified experimentally on a single cage gimbal setup that isolates one rotational degree of freedom. The drone stabilizes itself around the upright position, and successfully maintains a reference attitude.

1 INTRODUCTION

The capability of flying insects, bats and small birds to maneuver and navigate with ease inside cluttered and confined environments has inspired the development of insect-like flapping wing aerial vehicles (FWAVs). These vehicles produce thrust by mimicking the complex wing motion of their natural counterparts. Unsteady aerodynamic phenomena that occur in low Reynolds number flapping flight produce significantly higher lift values than fixed wing or rotary wing motion [1]. Advances in lightweight miniaturized electronics have enabled the construction of insect-like FWAVs ranging in size from the 102 g Flapper Drone [2] down to the 80 mg Robobee [3]. Like insects, the wing stroke motion takes place inside the body horizontal plane, which gives the FWAV the capacity to hover in place. Because of their small size, lightweight soft material wings and relatively low wing tip speeds, insect-like flyers are at lower risk of (causing) damage in the event of a crash. Instead, these FWAVs tend to bounce off of the object they collide with, after which they continue their flight [2, 4]. Also, the flapping wing motion makes the wings unlikely to tangle on loose objects like cables and plant leaves. The combination of hovering flight and collision resilience makes insect-like FWAVs especially well suited to be deployed inside confined and cluttered environments. Possible applications are the inspection of difficult to reach industrial installations even in small cavities, crop health monitoring inside greenhouses, and the exploration of partially collapsed buildings after a natural disaster.

The test vehicle is the KUlibrie, a tailless two-winged flapping wing micro air vehicle with passive wing inclination that is in development at KU Leuven [5, 6]. The vehicle is inspired by the propulsion principles of hummingbirds and insects. The KUlibrie fits within a group of tailless, two-winged insect-like FWAVs of similar size, generally with a 10-25 g body mass and 15-20 cm total wingspan [7]. Maintaining controlled flight for this group of vehicles is an ongoing challenge. Insect FWAV flight is inherently unstable [8, 9], so a flight controller needs to continuously perform small control actions to stabilize the vehicle. Also, the small size imposes stringent weight restrictions that impede the number of actuators and onboard sensors and the amount of computing power that can be installed. Some examples of FWAV prototypes successfully maintain free flight around hover [2, 3, 4, 5, 10]. However, further improvements are required in robustness against modeling inaccuracies or uncertainties and time-dependent flight characteristics before the technology is ready for use in real-world applications.

This work employs a free flight simulation environment to evaluate attitude estimation and flight control of an insect-like FWAV around hover. Attitude estimation and control are validated on an experimental setup, with the prototype drone mounted inside a gimbal system. The control architecture in this study is based on PID, because of its common use as a relatively simple yet effective technique with low computational load. The simulation environment and the gimbal setup will later serve as building blocks for the future development and evaluation of more advanced control techniques, which can then be compared to the PID controller performance.

*Email address: thomas.roelandt@kuleuven.be

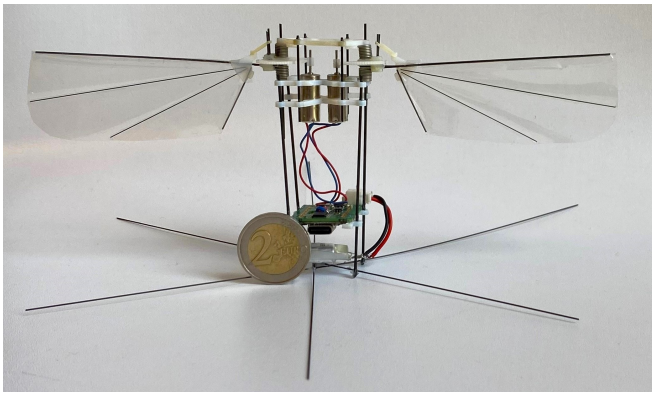


Figure 1: KUlibrie drone prototype, with a 2 euro coin added for size reference.

2 KULIBRIE FLAPPING WING SYSTEM

2.1 Design and key parameters

Figure 1 shows the latest KUlibrie prototype. The stroke motion of each wing is actuated independently by a brushless DC-motor, connected to the wing by a direct drive gear transmission. Two torsion springs are mounted at the root of the wing stroke axis. The spring stiffness is set such that the natural frequency of the flapping wing system is close to the stroke frequency, resulting in a resonant effect that improves driveline performance. Kinetic energy is stored in the springs as potential energy, and released again at the start of the subsequent stroke. The driveline design and performance are studied in detail in [5, 6]. The wing inclination motion, which is dominant in determining the value of the angle of attack, is passive. It is the result of the interaction between aerodynamic forces on the wing surface, counter-torque from a rubber band attached at the base of the wing, and inertial effects. This rubber band also initiates the pronation and supination at the end of upstroke and downstroke respectively, ensuring a positive angle of attack at the start of the next stroke [11].

Drone mass is 15.8 g and total wingspan is 195 mm. Table 1 lists key design parameters. The body frame is constructed with carbon fiber and 3D-printed acrylic resin components. The wings feature a 10 μm Mylar membrane, reinforced with a carbon fiber veined skeleton. The stroke frequency is set at 19 Hz. The current prototype has an onboard 'Seeed Studio XIAO nRF52840 Sense' processing unit with a built-in 3-axis gyroscope and 3-axis accelerometer IMU. A 100 mAh LiPo battery provides the system with sufficient energy for approximately 90 seconds of operation.

Compared to FWAV prototypes that use linkage mechanisms and one motor to drive the motion of both wings, the two decoupled motors on the KUlibrie provide greater control over its wing motion. The ability to alter the stroke amplitude and mean stroke angle for each wing individually theoretically provides sufficient controllability for free flight maneuvering without the need for additional drivers. Instead, flight can be realized by only control of the wing stroke motion. This pure wing stroke motion-based control strategy is similar to that of the Purdue Robotic Hummingbird [12] and the Robobee [3]. The simplicity of the direct drive stroke motion-based control strategy lends itself well for further vehicle down-scaling, and may be necessary for truly insect scale FWAV prototypes [3]. A challenge that comes with this control strategy is that the same motors responsible for providing vehicle thrust are also used to perform attitude control and velocity control. This means that some of the available payload capacity is sacrificed for flight maneuvering. Care must also be taken that the combined control signals do not saturate the motors. The control signals and prevention of motor saturation are discussed further in the next sections.

Parameter	Value	Unit
Total wingspan	195	mm
Single wing length	83.5	mm
Mean chord length	20	mm
Aspect ratio	4.175	-
y-distance CoG - CoR	10	mm
z-distance CoG - CoR	29.5	mm
Total body mass	15.8	g
Single wing mass	0.2	g

Table 1: Drone body and wing design parameters.

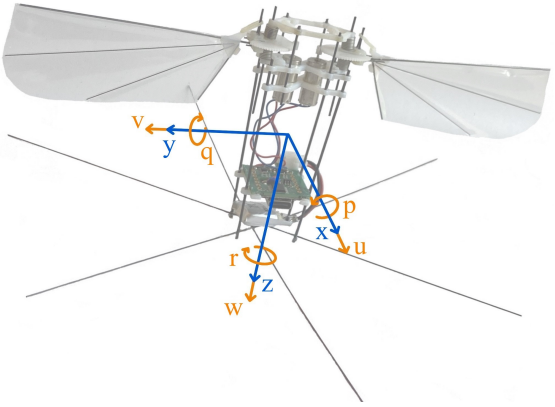


Figure 2: Body translation components $[u, v, w]$ and rotation components $[p, q, r]$, defined with respect to the body-fixed coordinate system xyz.

2.2 Flight simulation model

A discrete-time flight simulation model in MATLAB Simulink emulates the flapping wing drone prototype. Model input parameters include the drone dimensions, inertia (cfr. Table 2) and motor characteristics. Wing stroke motion is calculated based on the motor voltage inputs and wing dimensions [6]. Wing inclination and deviation motion are

http://www.imavs.org/

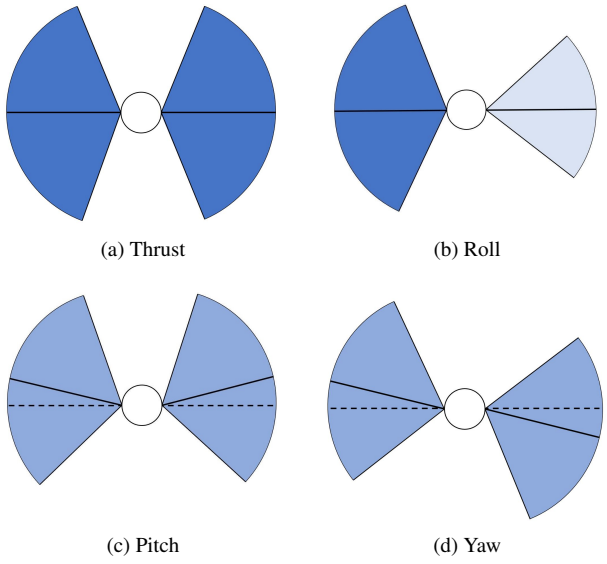
http://www.imavs.org/

described based on earlier experimental results [6]. Aerodynamic force production is computed using an experimentally validated rigid wing quasi-steady aerodynamics model [6, 13]. The discrete time step of the simulation is set to 1% of the stroke cycle period to capture the fast body oscillations that are present in two-winged FWAV systems, occurring at stroke frequency [7, 14]. Finally, body motion is computed using rigid body equations of motion. The drone body has three translation (u, v, w) and three rotation (p, q, r) degrees of freedom (DOFs), resulting in the dynamics model:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{V}(t)), \tag{1}$$

with $\mathbf{x}(t) = \begin{bmatrix} u(t) & v(t) & w(t) & p(t) & q(t) & r(t) \end{bmatrix}^\top$, as indicated in Figure 2, and $\mathbf{V}(t) = \begin{bmatrix} V_{\text{left}}(t) & V_{\text{right}}(t) \end{bmatrix}^\top$ the voltages applied to the left and right wing, respectively. The over-dot indicates a derivative with respect to time variable t. The explicit dependence on t is further omitted where it is implicitly clear. The reader is referred to [6, 14] for a detailed derivation of the dynamics model $\mathbf{f}$ in Equation 1.

2.3 *Control actions*



(a) Thrust (b) Roll

(c) Pitch (d) Yaw

Figure 3: Top view of control action effects on stroke amplitude and mean stroke angle (solid black line). Stroke amplitude determines the mean lift force. A darker shade indicates a higher lift force.

The voltages $\mathbf{V}$ are applied to the two motors in the form of block waves. The block waves for the left and right wing are set independently. A non-symmetric setting generates torque on the drone body. Specifically, the direct drive configuration controls both the stroke amplitude and the mean stroke angle for each wing separately through a square wave with variable amplitude and mean value:

$$V_{\text{left}}(t) = \begin{cases} V_{I0} + dV + V_0 & \tau < 0.5 + ds \\ -(V_{I0} + dV - V_0) & 0.5 + ds < \tau < 1 \end{cases} \tag{2a}$$

$$V_{\text{right}}(t) = \begin{cases} V_{I0} - dV + V_0 & \tau < 0.5 - ds \\ -(V_{I0} - dV - V_0) & 0.5 - ds < \tau < 1 \end{cases} \tag{2b}$$

$\tau = ft$ represents normalized time, ranging from $\tau = 0$ (start of upstroke) to $\tau = 1$ over one full stroke cycle. ds marks a deviation from the center of the time interval of a full stroke. The individual components of Equation 2 affect the motion of the vehicle in the following way (see also Figure 3):

Thrust: An increase/decrease in motor voltage amplitude V_{I0} results in an increase/decrease in lift production. If the increment in stroke amplitude for the two wings is equal, net torque is zero and only total lift production changes.

Roll: An asymmetric increment dV of the motor voltage amplitudes increases lift produced on one wing, and decreases lift on the other wing. This difference in lift production results in a net rolling torque.

Pitch: A shift in mean motor voltage V_0 results in a symmetric change in the mean stroke angle. This produces a longitudinal shift of the mean lift vector for each wing. The result is a net pitching torque.

Yaw: A shift ds in duration of upstroke versus downstroke also shifts the mean stroke angle. Because the wing stroke axes do not coincide with the drone center of gravity (CoG), the length of the moment arm between the aerodynamic drag vector on each wing and the drone CoG depends on the stroke angle. When an asymmetric shift in mean stroke angle is applied, this time-varying moment arm for the two wings results in a net yawing torque. For small alterations, the yawing torque scales approximately linearly with the shift in mean stroke angle [15].

The four control actions result in an underactuated flight system. The rolling motion is coupled with lateral translation, and the pitching motion is coupled with longitudinal translation. The driveline configuration also allows for control actions based on stroke frequency modulation. However, the principle of resonance support sets a number of constraints on system control. Deviation of the actuation frequency from the set resonance frequency affects both system efficiency and system response [6].

3 CONTROL METHODOLOGY

3.1 *Cascaded flight controller layout*

Figure 4 shows the controller layout incorporated into the free flight simulation. An outer-loop velocity controller computes the reference input for an inner-loop attitude controller, based on the desired flight velocity. The attitude controller

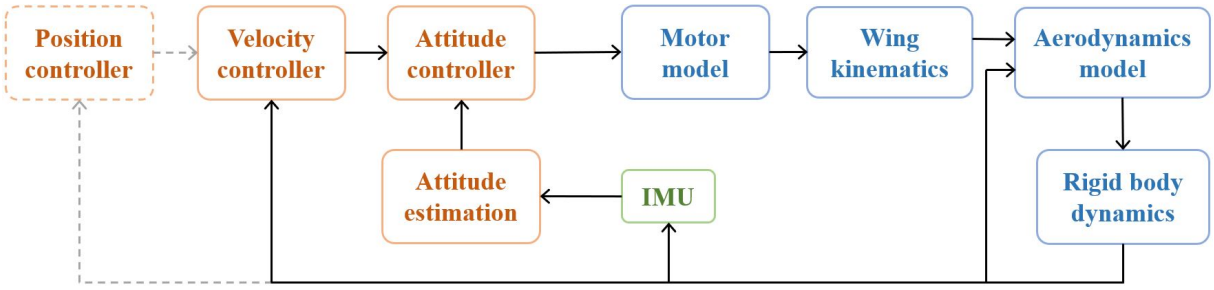


Figure 4: Schematic overview of the closed-loop control architecture in the flight simulation environment.

uses the output of an attitude estimation algorithm that is based on onboard IMU readings. The attitude estimation and control methods are discussed in the next sections. The cascaded flight controller may be expanded to be preceded by an additional outer-loop position controller that computes velocity controller inputs based on a reference flight trajectory. The expansion to include position control is highlighted in dashed lines in Figure 4. Since the onboard IMU does not measure the vehicle velocity, the velocity controller directly uses the actual velocity components $[u, v, w]$ of the vehicle. In future work, velocity estimation may be based on measurements from an external multi-camera configuration [3], or onboard camera system [16].

3.2 *Attitude estimation*

The attitude is estimated based on accelerometer and gyroscope readings. Assuming body accelerations $a_{x,y,z}$ are small in comparison to gravitational acceleration g, the accelerometer readings can be used to estimate the body roll and pitch angles:

$$\begin{cases} \Phi_{ac} = \arctan\left(\frac{a_y}{a_z}\right) \\ \Theta_{ac} = \arcsin\left(\frac{a_x}{g}\right) \end{cases} . \tag{3}$$

The gyroscope measures the velocities of the rotational DOFs, which are transformed from the body reference frame to a global reference frame according to:

$$\begin{cases} \dot{\Phi}_{gy} = p + (q \sin\Phi_{gy} + r\cos\Phi_{gy})\tan\Theta_{gy} \\ \dot{\Theta}_{gy} = q\cos\Phi_{gy} - r\sin\Phi_{gy} \\ \dot{\Psi}_{gy} = (q\sin\Phi_{gy} + r\cos\Phi_{gy})\sec\Theta_{gy} \end{cases} . \tag{4}$$

$[\Phi,\Theta,\Psi]$ respectively represent body roll, pitch, and yaw. Biases in the gyroscope measurements make its attitude estimation sensitive to drift. Gyroscope readings are modeled with additional Gaussian noise of variance 10^{-4} rad^2/s^2 and mean value (i.e. bias) of 0.3 rad/s. To limit the effect of bias on the gyroscope attitude estimation, the measurements are passed through a High-Pass Filter (HPF). The HPF excludes the DC biases from the gyroscope readings.

FWAVs are known to exhibit fast body pitching oscillations that occur at the stroke frequency [17, 10, 12]. These oscillations result in noisy accelerometer readings, that impede pitch angle estimation. A Low-Pass Filter (LPF) removes the effect of body pitching oscillations on the accelerometer readings. The LPF cutoff frequency is set to 1 Hz.

Figure 5 shows the attitude estimation strategies that are considered. The first strategy is a complementary filter, for which the attitude estimation is a weighted sum of the estimations based on the accelerometer and gyroscope readings. By varying the weighting factor α between 0 and 1, the estimation relies more heavily on the gyroscope or on the accelerometer. The complementary filter is considered because of its simplicity, resulting in a low computational load.

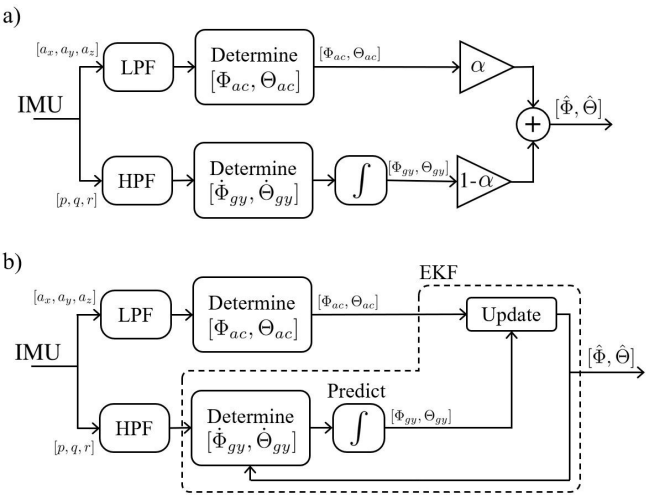


Figure 5: Schematic representation of attitude estimation methods: a) Complementary filter, b) Extended Kalman filter.

The second attitude estimation strategy uses an extended Kalman filter (EKF). The EKF is similar to the complementary filter, as its attitude estimation is based on a weighted average of the accelerometer and gyroscope readings. In this setting, the EKF uses the accelerometer estimates (Equation 3) to update its integrated gyroscope predictions (Equation 4). The main difference between the EKF and complementary filter, is that the EKF continuously updates the weight it attributes to each sensor reading based on the perceived process noise and measurement noise. Compared to

http://www.imavs.org/

http://www.imavs.org/

the complementary filter, the EKF is expected to be more robust to measurement noise and uncertainty, at the cost of a higher computational load.

Note that the attitude estimator only determines the estimated roll and pitch angle. Since the accelerometer readings are unaffected by the yaw angle, only the gyroscope readings provide information on the yaw rate. This means that the yaw rate measurement remains sensitive to sensing errors. Therefore, the real yaw angle is used as input for the yaw controller during the flight controller simulations. In future work, yaw sensing may be improved by the addition of an onboard magnetometer or camera.

3.3 Attitude control

The attitude controller has a PID architecture. PID is a popular control method in FWAV flight control [16, 10, 12] because of its simplicity and low computational load, which result in lower onboard memory requirements and allow for high sampling rates. Roll, pitch and yaw are controlled separately by three individual PID controllers, with the controller outputs corresponding to the components of Equation 2:

$$dV(t) = K_{P_1} e_\Phi(t) + K_{I_1} \int e_\Phi(t)\mathrm{d}t + K_{D_1} \frac{\mathrm{d}e_\Phi(t)}{\mathrm{d}t}, \quad (5a)$$

$$V_0(t) = K_{P_2} e_\Theta(t) + K_{I_2} \int e_\Theta(t)\mathrm{d}t + K_{D_2} \frac{\mathrm{d}e_\Theta(t)}{\mathrm{d}t}, \quad (5b)$$

$$ds(t) = K_{P_3} e_\Psi(t) + K_{I_3} \int e_\Psi(t)\mathrm{d}t + K_{D_3} \frac{\mathrm{d}e_\Psi(t)}{\mathrm{d}t}, \quad (5c)$$

where K_{P_i}, K_{I_i} and K_{D_i}, with $i = \{1, 2, 3\}$, are the respective PID gains. Moreover, the errors in the roll, pitch, and yaw angles are represented as $e_\Phi(t) = r_\Phi(t) - \Phi(t)$, $e_\Theta(t) = r_\Theta(t) - \Theta(t)$ and $e_\Psi(t) = r_\Psi(t) - \Psi(t)$, respectively. The reference signals r_Φ, r_Θ and r_Ψ are typically provided by the outer-loop velocity controller (see Figure 4). Finally, note that V_{I0} is not included in the attitude controller since it does not affect the orientation of the vehicle.

3.4 Determination of PID parameters

This section focuses on the setting of the PID gains of Equation 5. The gains are first obtained in an automated manner inside the flight simulation environment. The PID parameters are subsequently used as initial values for the experimental attitude controller, with further tuning being carried out manually.

A constrained nonlinear optimization problem is formulated to automate the tuning of the PID parameters. Specifically, the PID gains are iteratively adjusted based on a mean squared error objective over time span T_s. The gain tuning optimization problem is formulated around hover, i.e., $r_\Phi(t) = r_\Theta(t) = r_\Psi(t) = 0$, and is initialized with a roll and yaw angle of $\Phi = \Psi = 5°$ that represent an initial disturbance away from hovering conditions. The optimization problem is formally defined as:

$$\begin{aligned} \underset{K_{P_i}, K_{I_i}, K_{D_i}}{\text{minimize}} \quad & \frac{1}{T_s} \int_0^{T_s} (e_\Phi^2(t) + e_\Theta^2(t) + e_\Psi^2(t))\mathrm{d}t \\ \text{subject to} \quad & \\ \text{Eqs. } 1,2,3,4,5 \quad & \forall t \in [0, T_s], \\ K_{P_i}, K_{I_i}, K_{D_i} > 0 \quad & i = \{1, 2, 3\}. \end{aligned} \quad (6)$$

This formulation tunes all controller gains simultaneously, and easily re-tunes the controller in case a design alteration is made to the FWAV. The constraints in Equation 6 ensure that the PID gains are positive, and that the combined contribution of the PID outputs (according to Equation 2) does not exceed the maximum motor voltage $V_{max} = 3.4$ V. If at any time during simulation motor voltages V_{left} or V_{right} would exceed V_{max}, then voltage is made to saturate. Attitude estimation is achieved through the EKF. The problem is implemented in MATLAB using the built-in *fmincon* function with the interior-point algorithm. When the control system is extended to include velocity control, the optimization method in Equation 6 can be repeated to tune the velocity controller gains. In this case, the error terms are replaced by $e_u(t)$, $e_v(t)$, and $e_w(t)$.

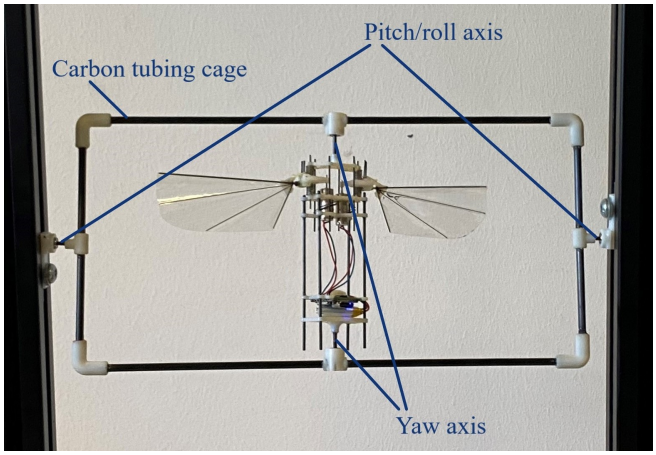


Figure 6: Single cage gimbal setup.

Moment of inertia (kgm^2)	Drone	Drone inside gimbal
I_{xx} (roll)	14.7×10^{-6}	75.6×10^{-6}
I_{yy} (pitch)	13.6×10^{-6}	74.5×10^{-6}
I_{zz} (yaw)	1.9×10^{-6}	2.1×10^{-6}

Table 2: Moments of inertia.

3.5 Experimental attitude controller setup

The real-life attitude controller tuning is done manually. To simplify the tuning process, each orientation angle is tuned

http://www.imavs.org/

separately by rotating the drone only about one axis at a time. For small corrections around hover, the three degrees of freedom in rotation can be considered to be uncoupled [9]. Roll, pitch and yaw are therefore controlled through three separately tuned PID controllers. The settings resulting from the simulated PID controller are used as a starting point for the physical controller tuning process.

Figure 6 shows the experimental attitude controller tuning setup. The drone is mounted inside a single cage gimbal with ball bearings. The inner rotation axis, the yaw axis, extends along the body vertical through the center of the drone. The outer rotation axis corresponds to pitch or roll, depending on the direction in which the drone is mounted. The outer rotation axis is positioned slightly below the drone center of gravity. This means that the upright position is an unstable equilibrium and the drone tends to flip over. Therefore, if the drone manages to remain upright during operation, this is only possible through a properly operating attitude controller. By addition of a fixture over the yaw axis, the yaw angle can be locked, thereby only leaving a pure rolling or pure pitching motion. Similarly, the outer rotation axis can be locked, only leaving yaw motion. The cage is a lightweight structure with 4mm carbon tubing. The cage material and dimensions are a compromise between a lightweight construction to minimize added inertia and a requirement of sufficient stiffness to avoid bending of the cage.

Table 2 compares the inertia for the free drone and the gimbal configuration. Compared to free flight the yaw motion is relatively unaffected, however there is a significant increase in the moments of inertia linked to pitch and roll. Rotational acceleration is lower when the drone is mounted inside the gimbal, giving the attitude control more time to respond to an error in orientation. Also, the increased moments of inertia imply that a larger counter torque is needed to counteract a pitch or roll velocity, thus inducing a larger control action. The gimbal structure is a test configuration setup to evaluate the effectiveness of the attitude estimation and control actions.

During the experiment, attitude estimation computations are performed on the onboard processing unit. The attitude estimation sampling rate is 200 Hz. State estimation data are sent to a measurement laptop via Bluetooth for a posteriori processing and analysis.

4 RESULTS

4.1 Attitude estimation strategies

The performance of the complementary filter and the EKF is compared through an example flight simulation of 10 s, with the drone released from a hovering condition. At $t = 2$ s the vehicle experiences a positive rolling torque disturbance, and at $t = 6$ s the vehicle experiences a positive pitching torque disturbance. Figure 7 shows the actual body roll and pitch angles, as well as the estimated values and estimation errors from the EKF and different complementary filter settings. The fast body pitching oscillations are clearly visible and have an amplitude of approximately 2.5°.

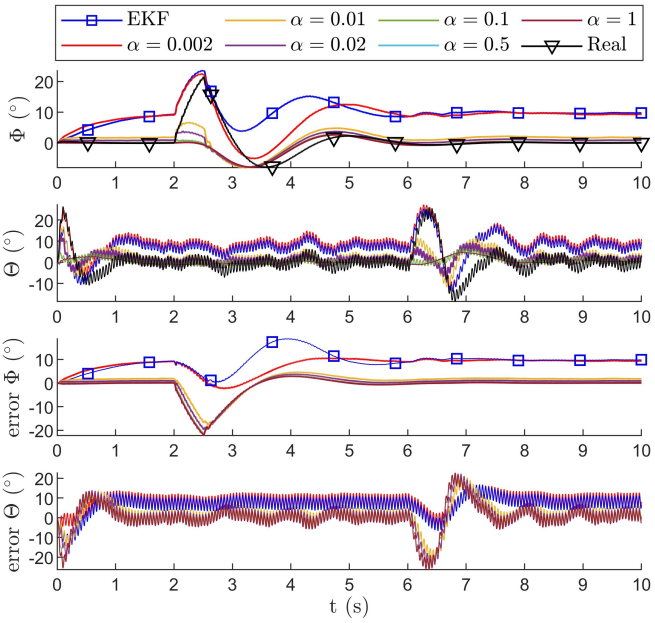


Figure 7: Comparison of roll (Φ) and pitch (Θ) estimation strategies.

- At very low values of α ($\alpha = 0.002$), the complementary filter relies heavily on the gyroscope readings for its attitude estimation. The estimator shows a good correlation at times when the system is perturbed. However, when the system is at rest, a bias term is present in the attitude estimation.
- When α increases, the complementary filter relies more on the accelerometer readings. This removes the bias on the attitude estimation when the system is at a steady state. However, the attitude estimator becomes less responsive to fast changes in orientation. For α in the range from 0.1 to 1, the attitude estimator behavior is approximately identical.
- There is a significant difference in attitude estimation performance between $\alpha = 0.002$ and $\alpha = 0.01$. An intermediate value for α may lead to an attitude estimator that provides a better compromise between accurate attitude estimation during system perturbation and a low bias when the system is at rest.
- The behavior of the EKF is similar to that of the low α complementary filter. When the system is in a steady state, a bias is present in the attitude estimation that is only slightly lower than that of the low α complementary filter. When the system is perturbed, the EKF has a faster response than the high α value complementary filter. For the pitching motion, with the accelerometer readings affected by body pitching oscillations, the EKF behavior is very similar to

the $\alpha = 0.002$ complementary filter. For roll estimation, the EKF appears to rely more heavily on accelerometer readings than the $\alpha = 0.002$ complementary filter, resulting in a more damped response to a roll perturbation.

The EKF is selected for further integration into the simulated and physical flight controller schemes. The EKF shows an increased responsiveness to pitching perturbations compared to high α complementary filters. Also, the EKF is expected to be more robust to general process noise and sensor noise. The bias that is present with the EKF is expected to be remedied when the attitude controller is preceded by a velocity controller.

4.2 Attitude and velocity control simulation results

The combination of velocity control and attitude control is initially tested without position control. Figure 8 shows two consecutive controlled flight maneuvers. The vehicle is released in hover, with a small initial roll perturbation. After $t = 5$ s, the vehicle is commanded to accelerate at a constant rate to a forward flight velocity of $u = 0.5$ m/s, measured in the body coordinate system, and then maintain this forward flight velocity until $t = 10$ s, after which it should decelerate back to hover. The other body velocity components need to remain at $v = w = 0$ m/s. At $t = 15$ s, the vehicle is commanded to accelerate at a constant rate to a lateral flight velocity of $v = 0.5$ m/s, and then maintain this velocity until the end of the simulation.

The vehicle successfully stabilizes to follow the reference velocity signal. The outer-loop velocity controller initiates the forward flight maneuver by altering the pitch angle reference input in order to tilt the thrust vector forward. Similarly, the lateral flight maneuver is initiated by adjusting the roll angle reference. The lateral velocity combined with a non-zero body pitch angle, results in a yawing torque. During initiation of the lateral flight maneuver, the yaw angle increases to a maximum deviation of 18°, after which the yaw controller manages to return the yaw angle to its reference value. The velocity controller accommodates for the bias in the pitch and roll estimation angles by properly adjusting the reference signals that are sent to the roll and pitch controllers.

4.3 Experimental validation

Figure 9 shows the results for the experimental roll and pitch control respectively. Each time the drone is commanded to initially maintain an upright orientation for 5 seconds. Next, the drone should tilt its attitude by 20° and maintain the new position. Figure 9 only shows the estimated roll and pitch values for this maneuver. The controller settles at a steady state error of approximately 4° for the estimated roll angle, and 6° for the estimated pitch angle. The steady-state errors in the experimental data may be explained by the lower integral gains and higher derivative gains on the physical system compared to the simulation (cfr. Table 3). The lower integral gains and higher derivative gains prevent large oscillations of the system, but at the cost of a longer settling time. The attitude controller experiment is repeated also for a step input, and for a ramp input tilt where the drone subsequently needs to return to an upright orientation. Figures related to the other experimental attitude control cases are available in Appendix A. These figures show that the controller is able to maintain hover, stabilize a severe step input, return from a tilted state to upright, and perform a yawing motion in either direction. Video footage of the attitude control experiments for roll, pitch, and yaw is available in the supplementary material.

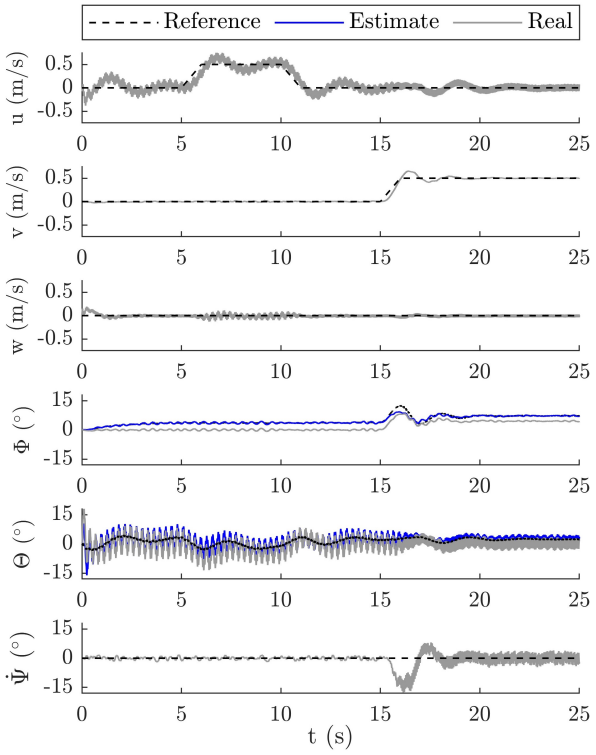


Figure 8: Velocity control during simulation flight maneuvering.

4.4 Study limitations

Flight simulation in this study assumes a system with known, time-independent flight characteristics that can be represented through rigid wing aerodynamics. In practice, the extent to which assumptions such as rigid wings affect model accuracy remains uncertain. Also, the dynamic characteristics of the FWAV may change due to design alterations or due to damage after a collision. The IMU readings may be influenced by additional sources of inaccuracies such as a variable bias, temperature fluctuations, or vibration rectification, increasing the measurement noise. Finally, motor heating is known to affect the system response to control actions, leading to time-varying dynamic characteristics. Future studies should consider uncertainties in the system dynamics, as

http://www.imavs.org/

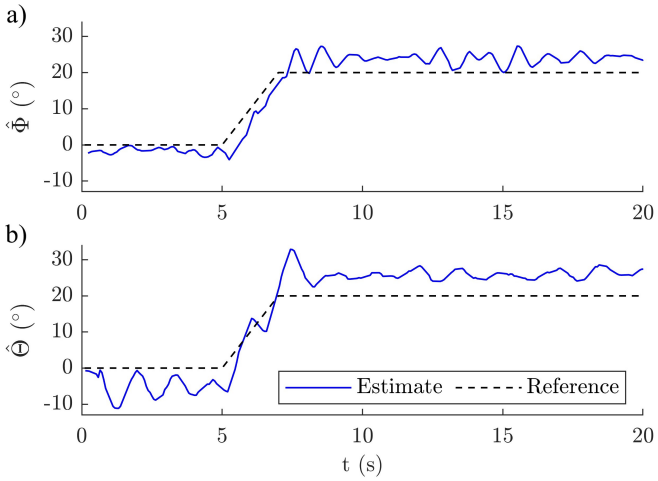


Figure 9: Attitude controller measurements for ramped inputs: a) roll estimation, b) pitch estimation.

well as their time-varying nature, for the implementation of a robust flight control strategy.

	Simulation			Experimental		
	K_P	K_I	K_D	K_P	K_I	K_D
Roll (Φ)	0.782	3.089	0.051	0.9	0.2	0.3
Pitch (Θ)	1.45	7.81	0.06	1.5	0.2	0.7
Yaw (Ψ)	0.02	0.045	0.003			
Yaw rate ($\dot{\Psi}$)				0.2	0.3	0.0
Unit	$\frac{V}{rad}$	$\frac{V}{rad \cdot s}$	$\frac{V}{rad/s}$	$\frac{V}{rad}$ $\frac{V}{rad/s}$	$\frac{V}{rad \cdot s}$ $\frac{V}{rad}$	$\frac{V}{rad/s}$ $\frac{V}{rad/s^2}$

Table 3: Control gains in free flight simulation and for experimental gimbal setup

5 CONCLUSION AND FUTURE WORK

This work discusses a control system for an insect-like tailless flapping wing drone. A simulation environment based on a quasi-steady aerodynamics model is used to compare attitude estimation techniques that employ onboard IMU readings. The attitude estimation is incorporated into a cascaded PID flight controller that stabilizes the drone attitude and velocity around hover. The outer-loop velocity controller successfully compensates for biases present in the attitude estimator. The attitude estimation and attitude control are evaluated experimentally with the drone mounted inside a single cage gimbal setup. The drone produces sufficient torque to stabilize its orientation around a desired tilt attitude. Because the gimbal setup significantly increases inertia, the controller gains and response time need to be reconfigured for free flight testing.

The relatively simple PID architecture is expected to suffice for unobstructed hovering flight when the flight system characteristics are static and accurately represented by the system model. In practice FWAVs experience conditions that may be challenging for a PID flight controller. The large range of flight modes and the uncertain, time-varying system characteristics encountered by physical FWAVs are expected to pose important additional challenges to the flight controller. Therefore, the flight simulation environment developed in this work will serve as a basis for future investigations that evaluate more advanced robust and adaptive control methods.

The controller is expected to derive advantage from additional sensing equipment to improve its state estimation capabilities. An onboard magnetometer would allow to determine the yaw angle, and camera readings would enable velocity and position control.

ACKNOWLEDGEMENTS

The Research Foundation - Flanders (FWO) is gratefully acknowledged for its support through research grant numbers 1S60222N and S001821N.

SUPPLEMENTARY MATERIAL

Video material of the experimental attitude estimation and control is available at `https://www.researchgate.net/publication/382950011_Supplementary_material_Video_footage_showing_rollpitchyaw_estimation_and_control_for_an_insect-like_flapping_wing_aerial_vehicle_inside_a_single_cage_gimbal_setup`.

REFERENCES

[1] D. D. Chin and D. Lentink. Flapping wing aerodynamics: from insects to vertebrates. *Journal of Experimental Biology*, 219:920–932, 2016.

[2] M. Karásek. Safe and collision-tolerant flying robots with flapping wings. Presented at the 13th international micro air vehicle conference and competition (IMAV2022). *url: https://www.youtube.com/watch?v=lq_2Ofh1x5M&t=13729s*, 2022.

[3] K. Ma, P. Chirarattananon, S. Fuller, and R. Wood. Controlled flight of a biologically inspired, insect-scale robot. *Science magazine*, 340, 2013.

[4] H. V. Phan and H. C. Park. Mechanisms of collision recovery in flying beetles and flapping-wing robots. *Science*, 370:1214–1219, 2020.

[5] F. Leys. *The Eurotrochilus Mechanicus: A robotic hummingbird driven by a resonant flapping mechanism*. PhD thesis, KU Leuven, Leuven, Belgium, 2017.

[6] S. Timmermans. *Flight performance of nano aerial vehicles with flapping wings: Experiment-based system*

http://www.imavs.org/

modelling. PhD thesis, KU Leuven, Leuven, Belgium, 2021.

[7] H.V. Phan and H.C. Park. Insect-inspired, tailless, hover-capable flapping-wing robots: Recent progress, challenges, and future directions. *Progress in Aerospace Sciences*, 111(100573), 2019.

[8] M. Karásek and A. Preumont. Flapping flight stability in hover: A comparison of various aerodynamic models. *International Journal of Micro Air Vehicles*, 4:203–226, 2012.

[9] K. Nguyen, L. T. K. Au, H. V. Phan, and H. C. Park. Comparative dynamic flight stability of insect-inspired flapping-wing micro air vehicles in hover: Longitudinal and lateral motions. *Aerospace Science and Technology*, 119(107085), 2021.

[10] H. V. Phan, S. Aurecianus, T. Kang, and H. C. Park. Kubeetle-s: An insect-like, tailless, hover-capable robot that can fly with a low-torque control mechanism. *International Journal of Micro Air Vehicles*, 11:1–10, 2019.

[11] T. Roelandt, T. Willems, F. Naets, and D. Vandepitte. High-speed camera measurement of insect-like flapping wing deformation using a speckle pattern wing membrane. In *14th International Micro Air Vehicle Conference and Competition (IMAV2023)*, 2023.

[12] Z. Tu, F. Fei, J. Zhang, and X. Y. Deng. An at-scale tailless flapping-wing hummingbird robot. i. design, optimization, and experimental validation. *IEEE Transactions on Robotics*, 36(5):1511–1525, 2020.

[13] Q. Wang, J.F.L. Goosen, and F. van Keulen. A predictive quasi-steady model of aerodynamic loads on flapping wings. *Journal of Fluid Mechanics*, 800:688–719, 2016.

[14] T. Roelandt and D. Vandepitte. Inherently stable descending flight of a tailless flapping wing micro air vehicle by upward wing elevation. *International Journal of Micro Air Vehciles*, 2023.

[15] S. Timmermans, F. Leys, and D. Vandepitte. Model-based evaluation of control roll, pitch, yaw moments for a robotic hummingbird. *Journal of Guidance, Control and Dynamics*, 40(11):2934–2940, 2017.

[16] D. Olejnik, M. Karásek, B. Duisterhof, K. Scheper, T. van Dijk, and G.C.H.E. de Croon. A tailless flapping wing mav performing monocular visual servoing tasks. In *11th International Micro Air Vehicle Conference and Competition (IMAV2019)*, pages 60–66, 2019.

[17] S. Armanini, M. Karásek, C. de Visser, G.C.H.E. de Croon, and M. Mulder. Flight testing and preliminary analysis for global system identification of ornithopter dynamics using on-board and off-board data. *AIAA Atmospheric Flight Mechanics Conference*, 2017.

APPENDIX A: ADDITIONAL EXPERIMENTAL ATTITUDE CONTROL CASES

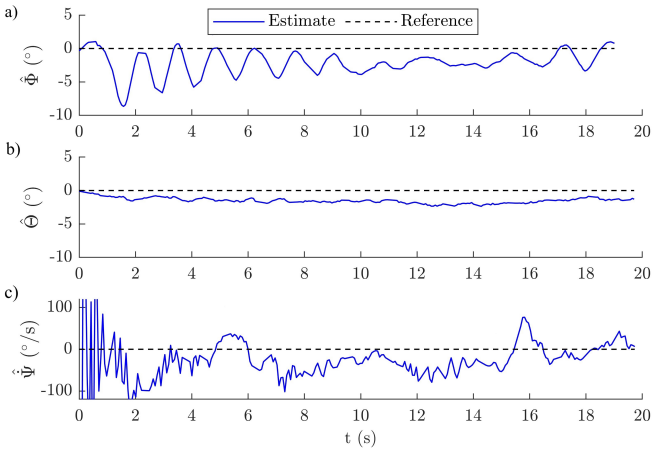


Figure 10: Attitude controller measurements for maintained hover.

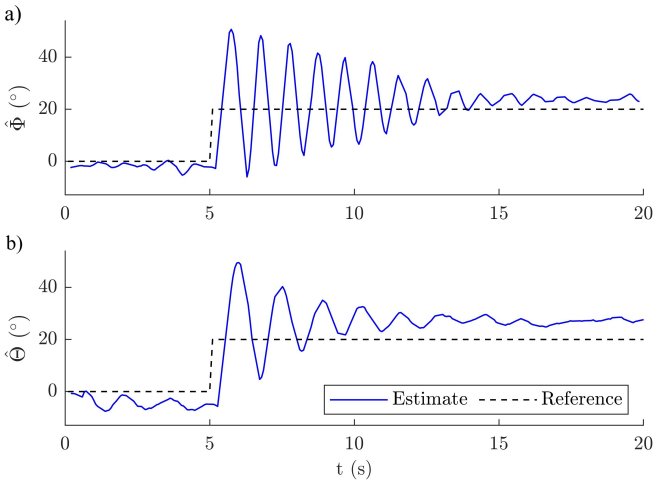


Figure 11: Attitude controller measurements for step inputs: a) roll estimation, b) pitch estimation.

http://www.imavs.org/

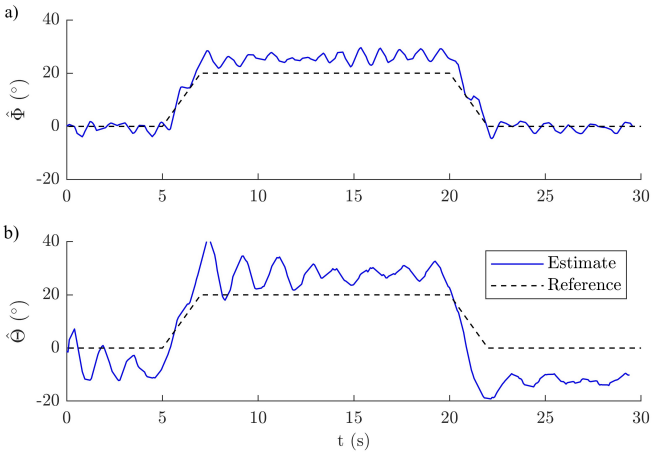


Figure 12: Attitude controller measurements for ramp inputs to 20° and back to upright: a) roll estimation, b) pitch estimation.

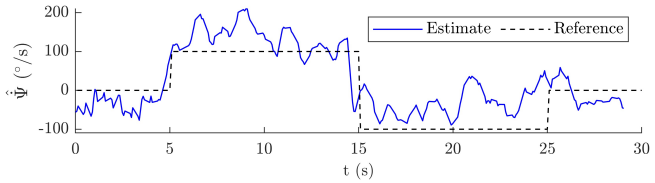


Figure 13: Attitude controller measurements for yaw rate step inputs.

http://www.imavs.org/

A Bio-Inspired Micro Air Vehicle and the Experimental Setup for Aerodynamic Testing

W.-L. Chan*, H.-Y. Chen, C.-L. Chin, and C.-H. Huang
National Cheng Kung University, No. 1, University Road, Tainan 70101, Taiwan

http://www.imavs.org/

ABSTRACT

This paper presents the design, construction, and preliminary testing of the bio-inspired micro air vehicle (MAV), CKopter-1. The MAV features a stable design with dual pairs of wings and a tail for flight control, achieving thrust via the clap-and-fling effect. Detailed geometrical parameters and weight distribution of the CKopter-1 are provided. An experimental setup combining a fan array wind system, a six-axis robotic arm, and a load cell was developed to further investigate the MAV's aerodynamic performance. Calibration of the fan array wind system and initial force measurements were conducted. The results highlighted the MAV's trimmed flight condition, although discrepancies between flight and fan array tests were noted. The findings indicate that while the current setup provides a systematic approach to assessing the aerodynamic performance of the CKopter-1, further refinements are necessary. The preliminary results pave the way for developing a comprehensive aerodynamic model for the MAV.

1 INTRODUCTION

Micro air vehicles (MAVs) have increasingly captivated researchers in recent years, driven by their potential in surveillance, environmental monitoring, and search-and-rescue operations, all while maintaining a compact form factor. Among the diverse MAV designs, bio-inspired MAVs stand out for their efficiency and maneuverability, drawing inspiration from the flight mechanics of birds and insects. Early efforts on bio-inspired flight focused on comprehending the dynamics of unsteady aerodynamics employed by flapping wings [1, 2, 3, 4, 5, 6, 7], with subsequent research transitioning towards applying this aerodynamic understanding to the development of functional flying prototypes. Over the past decade, the design and construction of bio-inspired MAVs have experienced exponential growth. While it is impossible to mention every single bio-inspired MAV prototype, several pioneering projects have stood the test of time and remained active for many years, such as the DelFly [8, 9], RoboBee [10, 11], KUBeetle [12, 13], and Colibri [14, 15]. Among these, DelFly has progressed from inherently stable prototypes to highly agile tailless prototypes [16]. Similarly, the NUS Flowerfly project [17, 18] has undergone a comparable transition.

While control and stability have been achieved in these prototypes, there remains a notable absence of comprehensive studies on the flight characteristics of these flying machines. Consequently, the design of bio-inspired MAVs often relies heavily on trial-and-error processes. Many prototypes are initially constructed without specific performance criteria in mind. A deeper understanding of flight characteristics would enable the integration of stability, controllability, and maneuverability considerations directly into the design process, facilitating the development of MAVs that meet defined requirements.

Over the years, attempts have been made to develop flight dynamic models of bio-inspired flying machines. For insect-inspired or hummingbird-like prototypes, some models are based on empirical aerodynamic models or computational fluid dynamics (CFD) simulations [19, 20, 21]. Due to the significant complexity of such studies, only a handful have included data measurement from actual prototypes [22, 23, 24]. Consequently, efforts to generalize the flight characteristics of these types of flying machines remain scarce.

In light of the discussion above, the primary objectives of this study are to design and fabricate a bio-inspired MAV as a testing platform to explore the potential for agile flight inherent in bio-inspired solutions, and to construct an experimental setup that systematically studies the aerodynamics and flight dynamics characteristics of such flying machines. This paper outlines the preliminary work towards achieving that goal.

This paper is structured as follows: Section 2 details the design and construction of the CKopter-1, including its flapping mechanism and wing fabrication. Section 3 describes the experimental setup and calibration process. Section 4 presents preliminary test results, and Section 5 concludes with a discussion on future work and potential improvements.

2 THE BIO-INSPIRED MAV

The bio-inspired MAV prototype, named the 'CKopter-1' is as shown in Figure 1. It is an inherently stable design with two pair of wings and a tail for stabilization and flight control, much like many other inherently stable flapping wing MAVs designed and flown around the World. The pairs of wings produces thrust by utilizing the clap-and-fling effect. Flight maneuvering is achieved by utilizing the elevator and rudder

*Email address(es): woeileong_chan@gs.ncku.edu.tw

http://www.imavs.org/

for pitch and yaw controls respectively. The relative position of the center of gravity (CG) and the aerodynamic centers (ACs) as show in Figure 2 makes the prototype very stable in the roll direction, hence roll control is not needed.

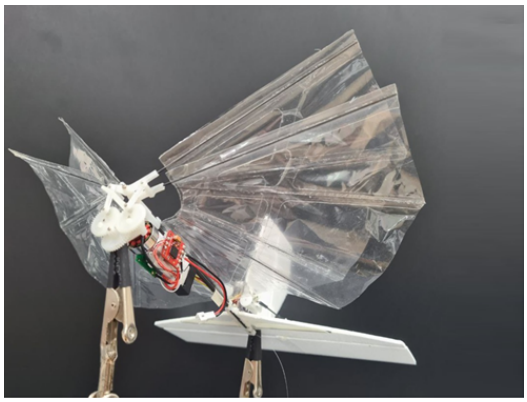


Figure 1: The bio-inspired MAV, CKopter-1.

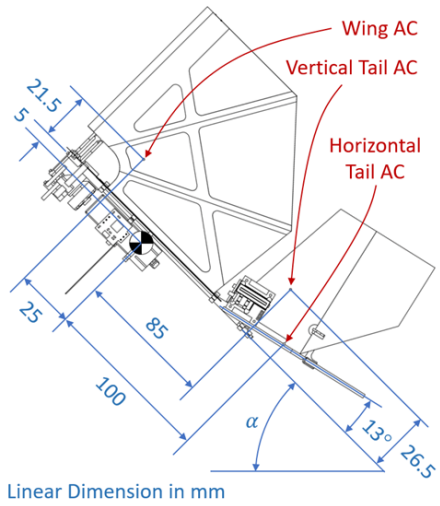


Figure 2: The estimated center of gravity and the aerodynamic centers.

The ACs shown in Figure 2 were estimated based on the geometrical parameters shown in Table 1. Since the prototype has four wings, it must be emphasized that the one-sided area of the wing is for one wing from root to tip. The data for the mean aerodynamic chord (MAC) was calculated using Equation 1, where S, $b/2$, c, and y stand for one-sided area, root-to-tip span, chord length, and distance in the span direction.

$$\bar{\bar{c}} = \frac{1}{S}\int_0^{\frac{b}{2}} c(y)^2\, dy \tag{1}$$

Based on the geometrical parameters, the tail volumes are 0.3327 and 0.2610 for the vertical and horizontal tails, respectively. The tail volumes were calculated using Equation 2 and Equation 3, where the subscripts v, h and w represent the vertical tail, horizontal tail, and the wing. In this context, the wing area S_w refers to the total area of a pair of wings from tip-to-tip. The symbol l, commonly referred as the arm length, denotes the longitudinal distance between the center of gravity (CG) and the AC of the respective tail surface.

	One-Sided Area (mm^2)	Root-to-Tip Span (mm)	MAC (mm)
Wing	8000	100	89.48
Horizontal Tail	4166	75	57.56
Vertical Tail	3157	63	50.67

Table 1: Geometrical parameters of the wing and tail.

$$\overline{V}_v = \frac{S_v l_v}{S_w b_w} \tag{2}$$

$$\overline{V}_h = \frac{S_h l_h}{S_w \bar{\bar{c}}_w} \tag{3}$$

The prototype has a total weight of 17.25gf. Figure 3 illustrates the weight distribution. Most of the weight comes from the 1 cell (1S) 130mAh lithium polymer (LiPo) battery, the two linear servos (LSM1300), the brushless motor (OP03X 20000Kv), the wings, and the flapping mechanism. The structural components, made primarily of 1.4mm $\times$ 1.4mm square carbon fiber rod, 1.5mm foam board, and Polyoxymethylene (POM), constitutes only 6% of the total mass. The lightweight avionics comprise the electronic speed controller (MX-3A) and the remote control receiver (DelTang 2.4GHz receiver).

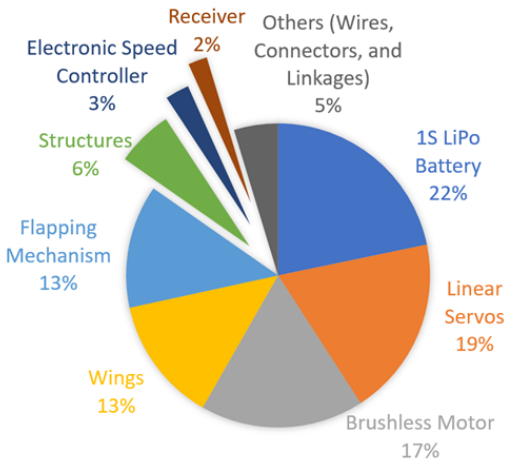


Figure 3: Weight distribution of CKopter-1.

http://www.imavs.org/

2.1 *Flapping Mechanism*

The CKopter-1 was designed to be a reliable platform for further investigation into bio-inspired flight dynamics. Therefore, a relatively conventional mechanism was chosen: a double crank-rocker mechanism, with each crank-rocker driving a pair of wings as shown in Figure 4. The 25.4:1 gear reduction ratio mechanism was adapted from an off-the-shelf mechanism rather than being proprietary. While the base structure and gears of the original mechanism were retained, significant modifications were made to the crank-rocker to enable clap-and-fling motions on the sides and top within a single flapping cycle as illustrated in Figure 5. These modified components were made from POM using an in-house computer numerical control (CNC) milling machine. The modification also allows each wing to have significantly larger stroke angle, which is 55° per wing.

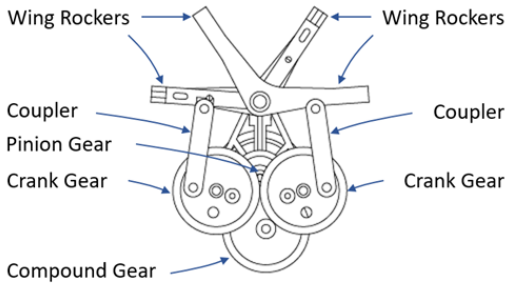


Figure 4: Flapping mechanism.

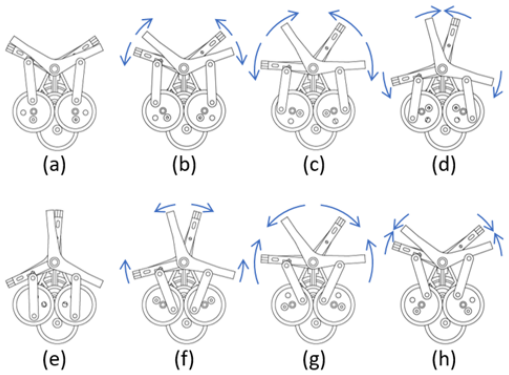


Figure 5: The flapping cycle.

2.2 *Wings*

The wing has a root chord of approximately 90mm and a tip chord of 50mm, with tapering starting at half the root-to-tip span. A 5° angle was added at the wing root, as depicted in Figure 6, to encourage wing rotation through wing twist deformation during flapping. It's worth noting that one side of the wing has a shorter chord than the other, differing by 3mm to accommodate the wing installation on the wing rocker. The wing membrane is made of 8μm transparent biaxially oriented polyethylene terephthalate (BoPET) film. The stiffeners, wing root, and leading edge are made of rounded carbon rods with diameters of 0.3mm, 0.6mm, and 0.8mm, respectively. These carbon rods were attached to the BoPET film using polyvinyl acetate-based adhesive, with transparent polyvinyl chloride (PVC) stickers added on top of the rods to reinforce the adhesion, as illustrated in Figure 7.

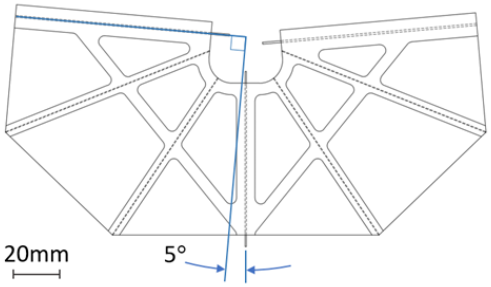


Figure 6: The working sequence of the flapping mechanism.

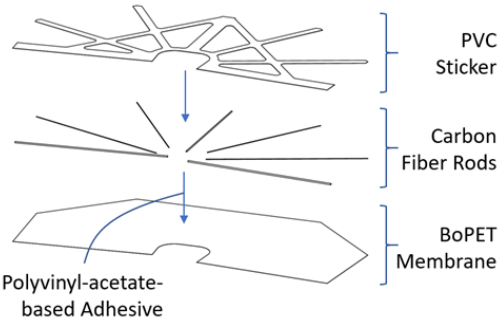


Figure 7: Fabrication of the wings.

2.3 *Trimmed Flight*

Remote piloted flight tests were conducted to assess the flight characteristics of CKopter-1, particularly to understand the trimming of control surfaces and to estimate the trim condition of the flight. Figure 8 shows a composite image extracted from high-speed video recording of a cruise flight. The trim condition was estimated using the video, revealing a cruise speed of approximately 1m/s and a flight incident angle (α) of about 43°. Furthermore, the flapping frequency was estimated to be 15.6Hz. It must be emphasized that achieving trimmed cruise flight in a confined space was extremely challenging. The estimated trim condition helped narrow down the testing range for the aerodynamic testing discussed in the subsequent section.

3 THE EXPERIMENTAL SETUP

As illustrated in Figure 9, the experimental setup for the aerodynamic test consists of a robotic arm holding a load cell

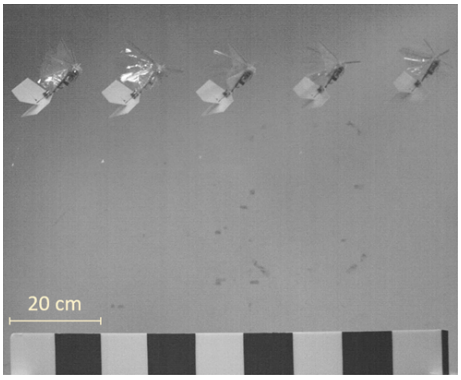


Figure 8: Flight test to estimate the trimmed condition of the prototype.

with the prototype attached. Incoming airflow is provided by a 5×5 fan array wind system. The robotic arm has six degrees of freedom (DoF), allowing users to control both transitional and rotational motion of the end effector. Specifically, the Xarm6 robotic arm is utilized, boasting a reach of 700mm, a travel speed of 1m/s, a rotational speed of 180°/s, and a maximum payload capacity of 5kg. The load cell used is the ATI Mini40, capable of measuring 3-axis forces and 3-axis torque. Such a setup promises future potential for dynamic testing.

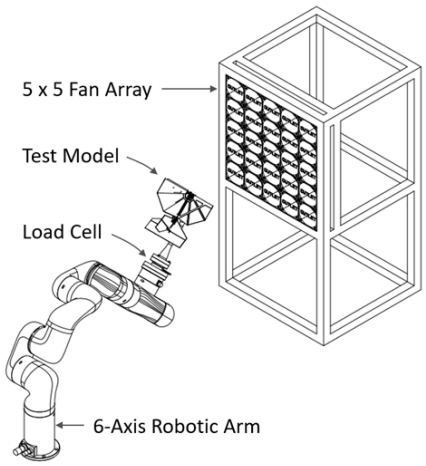


Figure 9: The experimental setup.

3.1 Fan Array Wind System

The design of the experimental setup was inspired by similar work using fan array systems [25]. The fan array wind system was assembled in-house using 25 high-performance cooling fans (Delta GFC0812DW-SM0113Y). Each fan features two counter-rotating blades and operates at 12V direct current (DC). With a maximum power consumption rated at 119.09W per fan and a maximum airflow of 5.015m^3/minute, the theoretical maximum wind speed is approximately 13m/s given the fan frontal area of 80×80mm. However, if all 25 fans were to run at maximum speed, they would collectively draw close to 3000W of power. The current DC power supply (GW Instek PSW 30-72) is only capable of supplying a maximum of 720W. Therefore, the current setup is not ready to operate at maximum speed, nor is it necessary to do so, as the trim flight speed of CKopter-1 is significantly below the maximum speed.

The overall system architecture of the fan array is depicted in Figure 10. Each fan is controlled via a pulse-width modulation (PWM) signal through an Arduino UNO, with each Arduino UNO managing five fans. A master Arduino UNO acts as the bridge, receiving user commands from a personal computer (PC) via a universal serial bus (USB) and sending commands to the other Arduinos to control individual fan speeds via the universal asynchronous receiver/transmitter (UART) protocol. All the Arduino codes were developed in-house.

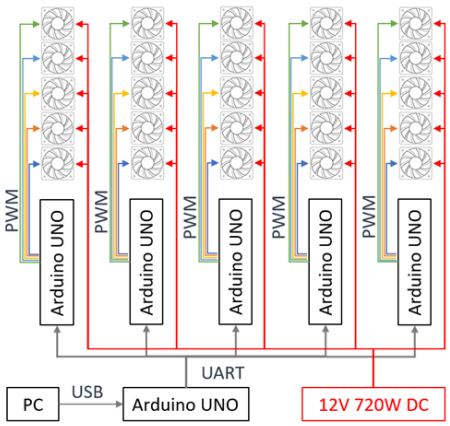


Figure 10: System architecture of the fan array.

The fan array system offers several advantages over traditional wind tunnels. The smaller fans have a faster reaction time, and when combined with independent control, they enable the generation of complex wind fields. The unbounded testing area reduces restrictions, opening up many possibilities for dynamic motion tests. For instance, in our case, we positioned a robotic arm downstream in the testing area. However, the system's downside is the significantly higher turbulence, which makes it less suitable for investigating delicate flow fields, though it may more accurately reflect the turbulent conditions of free flight.

In this study, the robotic arm was used to adjust the flight incident angle (α), as illustrated in Figure 11. The test model's rotational motion was centered on the CG position of the CKopter-1. While this setup was utilized for static testing, it can be extended to dynamic testing, where force and moment data are collected at specific pitch rates.

http://www.imavs.org/

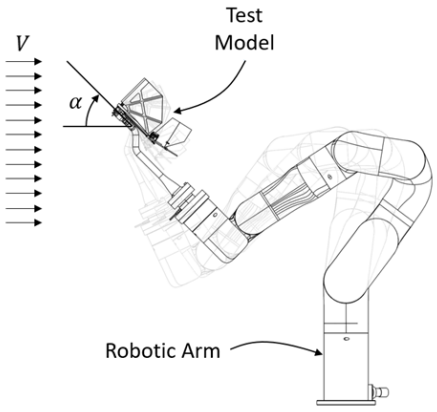


Figure 11: Robotic arm motion to achieve flight incident angle in the experiment.

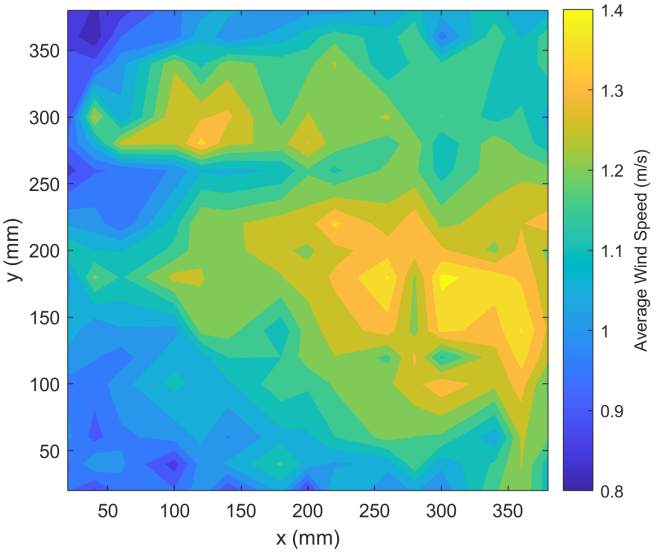


Figure 12: Average wind speed contour of the fan array at minimum speed setting.

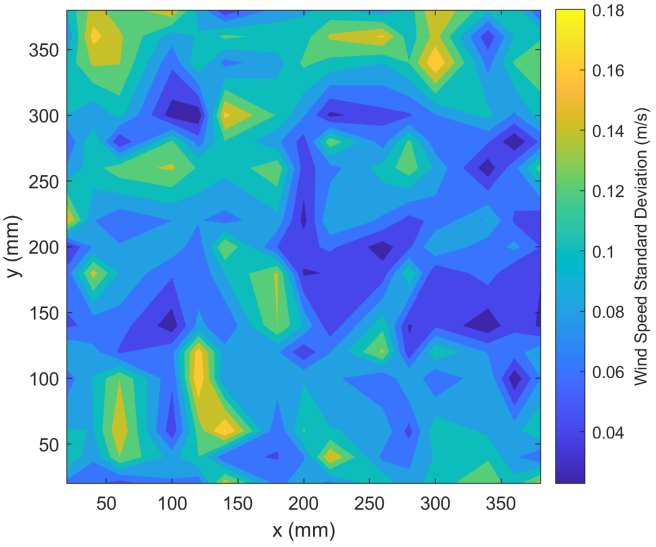


Figure 13: Wind speed standard deviation contour of the fan array at minimum speed setting.

4 CALIBRATION AND PRELIMINARY TEST RESULTS

4.1 *Fan Array Calibration*

The calibration of the fan array was conducted with the robotic arm holding a Benetech GT8911 hot wire anemometer, scanning through the entire 400×400mm area at 20mm intervals using a robotic arm. The target wind speed was approximately 1.1m/s, which corresponds to the estimated cruise speed of CKopter-1. An unforeseen challenge during the calibration process was the lower limit of the PWM signal to which the fans would respond. It was discovered that the fans do not respond to PWM signals lower than a 5% duty cycle. While the fans were intentionally chosen to be overpowered to accommodate higher speed tests in the future, the wind speed was still too high for testing CKopter-1 even at the minimum setting. Consequently, the fan array was positioned 2.8 meters away from the testing model to reduce the average wind speed.

The sampling rate of the wind speed measurement was 1Hz. A total of 10 data points were collected at each measurement point. The wind speed profile is depicted in Figure 12. The average wind speed was 1.14m/s, with the region at the center-right showing a slightly higher wind speed than the rest. Figure 13 depicts the standard deviation of the wind speed measurements. It is evident that the wind speed was very stable, with the highest standard deviation of 0.18m/s occurring at only a few spots. Further work is needed to improve the wind speed distribution for more accurate test results.

4.2 *Preliminary Force Measurement*

Using the attained wind profile, preliminary tests were conducted to determine the trim flight condition using a CKopter-1 test model, which closely resembles the flying prototype. The actual setup is depicted in Figure 14. Tests were conducted at flapping frequencies of 16Hz, with flight incident angles (α) ranging from 40° to 60° at 5° intervals. Tests were performed three times to ensure repeatability.

The cycle-averaged vertical force (positive upward), horizontal force (positive rearward), and pitching moment about the CG (positive pitch-up) are plotted in Figures 15, 16, and 17, respectively. The error bars represent standard deviation. Cycle-averaged force/moment refers to the average measurements over multiple continuous flapping cycles. Both forces exhibit an approximately linear increase with the incident angle. Notably, the horizontal force is the net result of forward thrust and aerodynamic drag. At an incident angle of 56.43°, marked by the red circle in the figures, the forward thrust and

http://www.imavs.org/

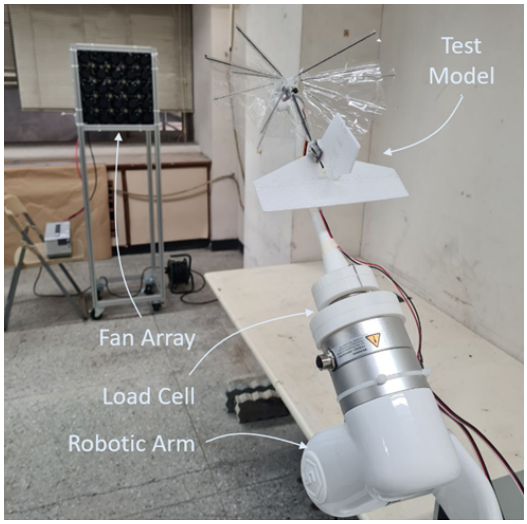


Figure 14: Actual setup in the laboratory.

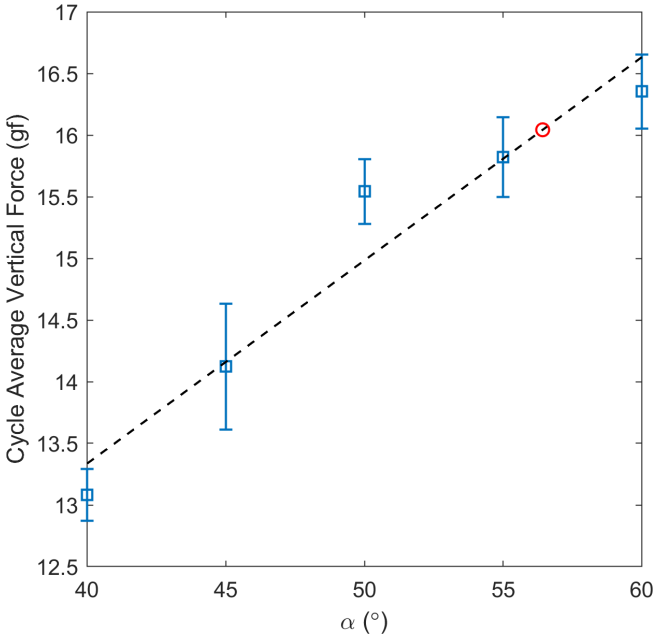


Figure 15: Vertical force at 1.14m/s wind speed and 16Hz flapping frequency.

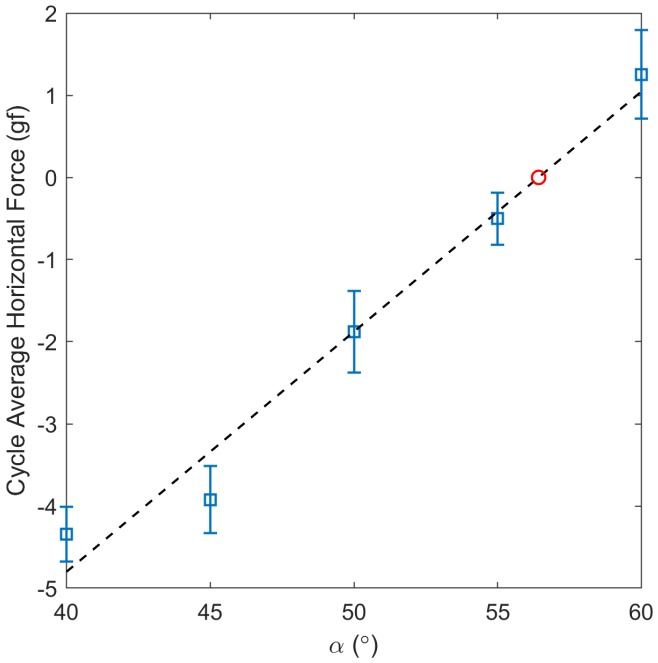


Figure 16: Horizontal force at 1.14m/s wind speed and 16Hz flapping frequency.

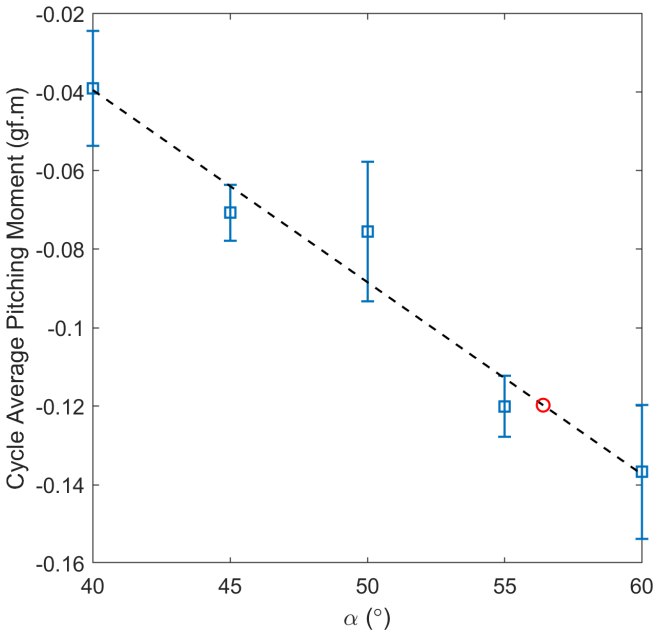


Figure 17: Pitching moment at 1.14m/s wind speed and 16Hz flapping frequency.

aerodynamic drag cancel each other out, indicating a trimmed flight condition where the prototype flies at a constant velocity. However, the vertical force at that point was 16.04 gf, which is less than the total weight of the flying prototype, 17.25 gf, as mentioned in Section 2.

Moreover, while the negative slope of the pitching moment plot confirms the prototype's static stability, it is clear that the pitching moment remains negative within the range of incident angles measured. Further tests are required to identify the trim condition that satisfies the criteria of zero pitching moment, zero horizontal force, and a vertical force of 17.25 gf. It is apparent that the trim condition estimated from the flight test cannot be fully reproduced in the fan array test. Several factors may contribute to this discrepancy, such as increased drag caused by the sting attachment of the test

http://www.imavs.org/

model, slight differences between the flying prototype and the force measurement test model due to fabrication tolerances, and inaccuracies in pitching moment measurement caused by the load cell's lack of precision and its position far from the CG due to size constraints.

Despite these challenges, the experimental setup provides a systematic way to assess the aerodynamic performance of the CKopter-1 and holds potential for dynamic experiments.

5 CONCLUSION

We have built an inherently stable bio-inspired MAV and performed remote control flight tests to estimate its trim condition. To further understand its aerodynamic performance, we designed and constructed an experimental setup to facilitate dynamic modeling. This setup combines a fan array wind system, a six-axis robotic arm, and a load cell, allowing us to measure the forces and moments acting on the MAV during various orientation. Although more work is needed to perfect the setup, our preliminary results are promising and indicate that a comprehensive aerodynamic model can be developed in the future. Future work includes improvement on the fan array calibration, utilizing smaller and more precise load cells, conduct dynamic test, and possibly incorporating a motion capture system to accurately track the test model's motion.

ACKNOWLEDGEMENTS

This project was supported by the National Science and Technology Council of Taiwan (112-2222-E-006-001-MY3).

REFERENCES

[1] H. Liu, C.P. Ellington, K. Kawachi, C. van den Berg, and A.P. Willmott. A computational fluid dynamic study of hawkmoth hovering. *Journal of Experimental Biology*, 201(4):461–477, 1998.

[2] M.H. Dickinson, F.O. Lehmann, and S.P. Sane. Wing rotation and the aerodynamic basis of insect flight. *Science*, 284(5422):1954–1960, 1999.

[3] Z.J. Wang. Vortex shedding and frequency selection in flapping flight. *Journal of Fluid Mechanics*, 410:323–341, 2000.

[4] R. Ramamurti and W. Sandberg. A three-dimensional computational study of the aerodynamic mechanisms of insect flight. *Journal of Experimental Biology*, 205(10):1507–1518, 2002.

[5] S.P. Sane. The aerodynamics of insect flight. *Journal of Experimental Biology*, 206(23):4191–4208, 2004.

[6] Z.J. Wang, J.M. Birch, and M.H. Dickinson. Unsteady forces and flows in low reynolds number hovering flight: Two-dimensional computations vs robotic wing experiments. *Journal of Experimental Biology*, 207(3):449–460, 2004.

[7] F.O. Lehmann, S.P. Sane, and M.H. Dickinson. The aerodynamic effects of wing-wing interaction in flapping insect wings. *Journal of Experimental Biology*, 208(16):3075–3092, 2005.

[8] G. de Croon, K.M.E. de Clerq, R. Ruijsink, B. Remes, and C. De Wagter. Design, aerodynamics, and vision-based control of the delfly. *International Journal of Micro Air Vehicles*, 1(2):71–97, 2009.

[9] C. De Wagter, S. Tijmons, B.D.W. Remes, and G.C.H.E. de Croon. Autonomous flight of a 20-gram flapping wing mav with a 4-gram onboard stereo vision system. In *IEEE Conference on Robotics and Automation (ICRA)*, 2014.

[10] K.Y. Ma, P. Chirarattananon, S.B. Fuller, and R.J. Wood. Controlled flight of a biologically inspired insect-scale robot. *Science*, 340(6132):603–607, 2013.

[11] E.F. Helbling, S.B. Fuller, and R.J. Wood. *Altitude estimation and control of an insect-scale robot with an onboard proximity sensor*, pages 57–69. Springer International Publishing, 2018.

[12] H.V. Phan, T. Kang, and H.C. Park. Design and stable flight of a 21g insect-like tailless flapping wing micro air vehicle with angular rates feedback control. *Bioinspiration & Biomimetics*, 12(3), 2017.

[13] H.V. Phan, S. Aurecianus, T. Kang, and H.C. Park. Kubeetle-s: An insect-like, tailless, hover-capable robot that can fly with a low-torque control mechanism. *International Journal of Micro Air Vehicles*, 11(3), 2019.

[14] A. Roshanbin, H. Altartouri, M. Karasek, and A. Preumont. Colibri: A hovering flapping twin-wing robot. *International Journal of Micro Air Vehicles*, 9(4):270–282, 2017.

[15] A. Preumont, H. Wang, S. Kang, K. Wang, and A. Roshanbin. A note on the electromechanical design of a robotic hummingbird. *Actuators*, 10(3), 2021.

[16] M. Karásek, F.T. Muijres, C. De Wagter, B.D.W. Remes, and G.C.H.E. de Croon. A tailless aerial robotic flapper reveals that flies use torque coupling in rapid banked turns. *Science*, 361(6407):1089–1094, 2018.

[17] Q.V. Nguyen, W.L. Chan, and M. Debiasi. Experimental investigation of wing flexibility on force generation of a hovering flapping wing micro air vehicle with double wing clap-and-fling effects. *International Journal of Micro Air Vehicles*, 9(3):187–197, 2017.

[18] Q.V. Nguyen and W.L. Chan. Development and flight performance of a biological-inspired tailless flapping

http://www.imavs.org/

wing micro air vehicle with wing stroke plane modulation. *Bioinspiration & Biomimetics*, 14(1):016015, 2018.

[19] M. Sun. Insect flight dynamics: stability and control. *Reviews of Modern Physics*, 86:615–646, 2014.

[20] S. Biswal, M. Mignolet, and A.A. Rodriguez. Modeling and control of flapping wing micro aerial vehicles. *Bioinspiration & Biomimetics*, 14(2):026004, 2019.

[21] B. Singh, K.A. Ahmad, M. Murugaiah, N. Yidris, A.A. Basri, and R. Pai. Quasi-steady aerodynamic modeling and dynamic stability of mosquito-inspired flapping wing pico aerial vehicle. *Frontiers in Robotics and AI*, 11, 2024.

[22] F.Y. Hsiao, T.M. Yang, and W.C. Lu. Dynamics of flapping-wing mavs: application to the tamkang golden snitch. *Journal of Applied Science and Engineering*, 15(3):227–238, 2012.

[23] K.M. Kajak, M. Karásek, Q.P. Chu, and G.C.H.E. de Croon. A minimal longitudinal dynamic model of a tailless flapping wing robot for control design. *Bioinspiration & Biomimetics*, 14(4):046008, 2019.

[24] R. McGill, N.-S.P. Hyun, and R.J. Wood. Modeling and control of flapping-wing micro-aerial vehicles with harmonic sinusoids. *IEEE Robotics and Automation Letters*, 7(2):746–753, 2022.

[25] D.A. Olejnik, S. Wang, J. Dupeyroux, S. Stroobants, M. Karasek, C. De Wagter, and G. de Croon. An experimental study of wind resistance and power consumption in mavs with a low-speed multi-fan wind system. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2989–2994, 2022.

http://www.imavs.org/

Bumblebees Exhibit Adaptive Flapping Responses to Air Disturbances

T. Jakobi*, S. Watkins**, A. Fisher**, and S. Ravi*
University of New South Wales and **RMIT University

http://www.imavs.org/

ABSTRACT

Insects excel in trajectory and attitude handling during flight, yet the specific kinematic behaviours they use for maintaining stability in air disturbances are not fully understood. This study investigates the adaptive strategies of bumblebees when exposed to gust disturbances directed from three different angles within a plane cross-sectional to their flight path. By analyzing characteristic wing motions during gust traversal, we aim to uncover the mechanisms that enable bumblebees to maintain control in unsteady environments. We utilised high-speed cameras to capture detailed flight paths, allowing us to extract dynamic information. Our results reveal that bees make differential bilateral kinematic adjustments based on gust direction: sideward gusts elicit posterior shifts in the wing closest to the gust, while upward gusts trigger coordinated posterior shifts in both wings. Downward gusts prompted broader flapping and increased flapping frequencies, along with variations in flap timing and sweep angle. Stroke sweep angle was a primary factor influencing recovery responses, coupled with motion around the flap axis. The adaptive behaviours strategically position the wings to optimize gust reception and enhance wing-generated forces. These strategies can be distilled into specific behavioural patterns for analytical modelling to inform the design of robotic flyers. We observed a characteristic posterior shift of wings when particular counteractive manoeuvres were required. This adjustment reduced the portion of the stroke during which the wing receiving gust forces was positioned in front of the centre of gravity, potentially enhancing manoeuvrability and enabling more effective recovery manoeuvres. These findings deepen our understanding of insect flight dynamics and offer promising strategies for enhancing the stability and manoeuvrability of MAVs in turbulent environments.

*Email address(es): t.jakobi@unsw.edu.au

1 INTRODUCTION

Insects have mastered the art of nimble control over their trajectory and attitude during flight, navigating both natural and urban environments with remarkable precision. These flight characteristics have inspired extensive studies into flapping wings, advancing our understanding of essential flight concepts such as unsteady lift mechanisms [1, 2, 3], control capabilities [4, 5], stability [6, 7], and underlying wing functions [8, 9, 10]. Research on these aspects has provided intelligent inspiration for robotic design [11], significantly contributing to the development of Micro Air Vehicles (MAVs).

For smaller-scale flying systems, complex airflow that is highly variable in strength and structure exists ubiquitously and impedes their control performance [12]. Even away from local effects such as wakes of structures and vegetation, wind closer to the ground is highly turbulent due to surface roughness in the atmospheric boundary layer (ABL) [13]. Animals, particularly insects, use flapping wings to produce the aerodynamic forces necessary for flight. Unsteady wing-air mechanisms such as dynamic stall and wake capture enable flapping wings to operate at low velocities, permitting precise control manoeuvres while hovering. Flapping wings at the low Reynolds numbers relevant to this scale also show positive effects in turbulence, aided by leading-edge vortex (LEV) formation [14], and has been shown to help overcome the effects of vortices [15, 16] and gusts [17, 18].

Within the demands of insect and MAV flight, the most aerodynamically challenging events are likely the step changes between air scenarios, particularly object approaches, which involve traversing shear layers and structured flows near surfaces. The scale of these local air phenomena are often close to a few characteristic insect-wing dimensions, making these transitions especially demanding. For a flying insect or MAV navigating through one of these wakes, the adverse flow interacting with the wings can be adequately described by a discrete gust containing local flow predominantly in a singular direction. From this, it can be understood that isolated gusts are an unavoidable element of the flight conditions at smaller scales.

Understanding the interplay between normal flapping forward flight and disturbance responses could provide valuable insights into the mechanical effects of certain s during flight, simplifying the situation by reducing it to more basic models. Recent studies demonstrate that organized voluntary body orientation manoeuvres accompany the translatory motions observed among many flying insects during turning,

take-off, and landing. They also indicate the average force periods incident upon the wings, reflecting the higher-order intentions of the bee through the wing outputs, which, when observed instantaneously, can appear as disorganised, chaotic data. During disturbances, the interaction of forces and wing motion become orders more complex, so understanding the average force periods incident upon the wings over longer duration (rather than instantaneous snapshots) could yield more helpful during a disturbance. Simplified models for the role of wing motion during disturbance negotiation could be found by investigating the top-down and motion effects displayed in high-speed experimental results, which is the focus of this study.

We extracted the kinematics of bumblebee wings during free flight traversal through strong gusts from three orthogonal directions. By analyzing the rotation angles of the wings, we aimed to understand how wing motion adapts to the direction of the gust incident upon the bee. This analysis provides valuable insights into how flapping wing motions in disturbances can effectively counteract adverse conditions, thereby maintaining stable flight in harsh outdoor airflows. Such understanding enhances our knowledge of insect flight mechanics and could inform the design of more resilient bio-inspired flying robots.

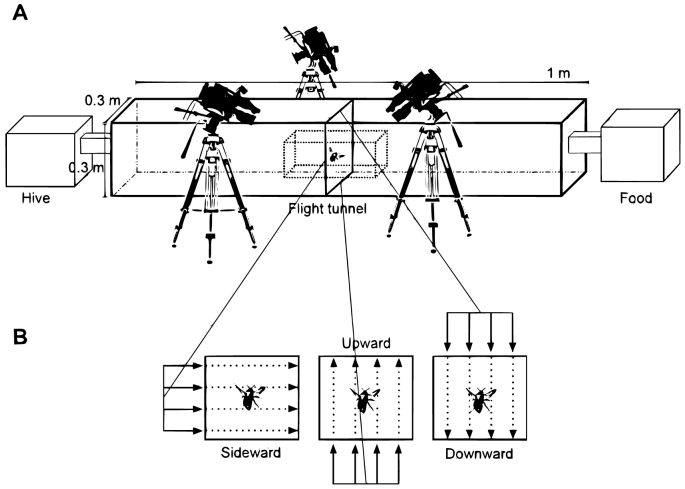


Figure 1: Experimental setup for bumblebee flight in gust disturbances. The laboratory environment for bumblebees contained a hive area (left), an interrogation flight tunnel (middle) and a feeding chamber (right). Gusts were made to flow across the cross-section of the interrogation region (B).

2 METHODOLOGY

2.1 Biological Specimen

Bumblebees are exemplary subjects for studying flight dynamics due to their advanced flapping control capabilities. They perform rapid manoeuvres, utilize sophisticated sensory and control strategies, carry substantial payloads, and function effectively in various weather conditions. Bumblebees are known for their adaptability and resilience, and are proficient learners and navigators, often traveling long distances for food. Their ability to adapt to new environments and execute consistent foraging missions makes them ideal for laboratory observation.

2.2 Laboratory Environment

The bumblebees species used in these experiments (*Bombus ignitus*) were maintained under controlled laboratory conditions as shown in Fig. 1A. The bees were sourced from a commercial breeder (Mini Polblack, Koppert, Arysta LifeScience Asia, Japan) as a single hive containing approximately 100 bees. From this hive, around 50 healthy foraging bees were identified and selected for the experiments. Data collected from bees in the same experimental setup as described here are published in [19]. More details on the experimental setup can be obtained by referring to that article.

The selected bees were anesthetized for the attachment of trackable markers. They were housed in a plastic cube measuring approximately 30 cm in each dimension. During a four-day habituation period, the bees were gradually acclimated to the experimental environment. Foraging bees were lured with a sucrose solution with a 1:4 sugar-to-water ratio (by weight) into the flight tunnel where bees could be interrogated (Fig. 1). They were trained to make repeated, consistent trips through the tunnel between the hive and the food source in this setup. The food sources were replenished at the end of each day to ensure continuous traffic to and from the hive. Trained bees flew directly from one end of the tunnel to the other, transporting sugar and pollen loads from the food chamber. Although new bees occasionally made flights through the tunnel, it was evident when a bee was familiar with the environment. Only these learned bees were selected for experimentation.

Bees were anesthetized by placing them individually into a refrigerator for up to 45 minutes. Once anesthetized, small paper markers were affixed to the dorsal surface of the thorax using a combination of wax adhesive and cyanoacrylate glue. After the adhesive dried, the bees were returned to the hive, where they resumed their routine behaviour within minutes. The bees appeared unaffected by the anesthetization process and the thorax-affixed trackable markers.

2.3 Implementation of Gust Disturbances

To study the bees' responses, we designed a flight tunnel with a controlled gust disturbance at the midpoint. The disturbance was generated using an air pump connected to an air knife nozzle, which produced a high-speed, 2D air jet across the tunnel's cross-section which challenged the bees without causing them to lose control. The gust velocity was set at 5 $\mathrm{m\,s^{-1}}$, determined by averaging the jet region of the flow between the two shear layers visible in Fig. 3. A visual representation of the gust disturbance within the tunnel, captured using Particle Image Velocimetry (PIV) is provided in

http://www.imavs.org/

Fig. 2. This technique allowed us to map the gust velocity and flow pattern over time, ensuring that the disturbance was consistent across experiments.

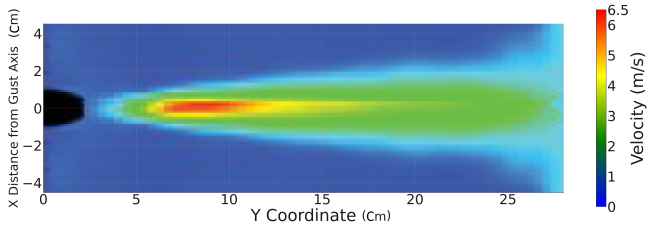


Figure 2: PIV image of the gust flow in the tunnel, showing velocities in the middle cross-section of the gust sheet mapped by colour. The example displays a gust oriented sideways with the camera frame looking down on the slice represented in the XY plane. The gust traverses a vertical cross-section (YZ) of the tunnel, moving from left to right over a period of 1 second (an average of 300 frames).

2.4 3D Localisation and Coordinate Systems

Table 1: Coordinate System Labels and Positive Directions.

Label	Positive Direction
X_G	Forward direction of the tunnel (facing hive side)
Y_G	Left-side of the tunnel (facing hive side)
Z_G	Top of the tunnel
X_{BF}	Anterior direction
Y_{BF}	Left-wing lateral direction
Z_{BF}	Dorsal direction
$Roll$	Right wing down
$Pitch$	Nose up
Yaw	Clockwise looking down

We used three Photron high-speed cameras (recording at 2000 fps with a shutter speed of 1/5000) positioned strategically around the tunnel to capture the bees' wing and body motions. To track body kinematics during flight, markers with three black dots in a triangular formation were affixed to the dorsal surface of the bumblebees' thoraxes (Fig. 3A). The markers were laser-printed on 80 gsm paper, and the black dots on a white background facilitated subsequent digitization using automated image-feature tracking software [20]. For wing motion tracking, four distinct anatomical landmarks on each wing were identified and tracked across at least two camera views per frame using automated and manual image feature extraction and part recognition algorithms. A 4th order Butterworth filter, chosen for its sharp roll-off characteristics, was applied post-digitization with separate cutoff frequencies tailored to wing and body data. This approach effectively removed noise while preserving the integrity of the recorded motion. The accuracy of automated tracking was validated

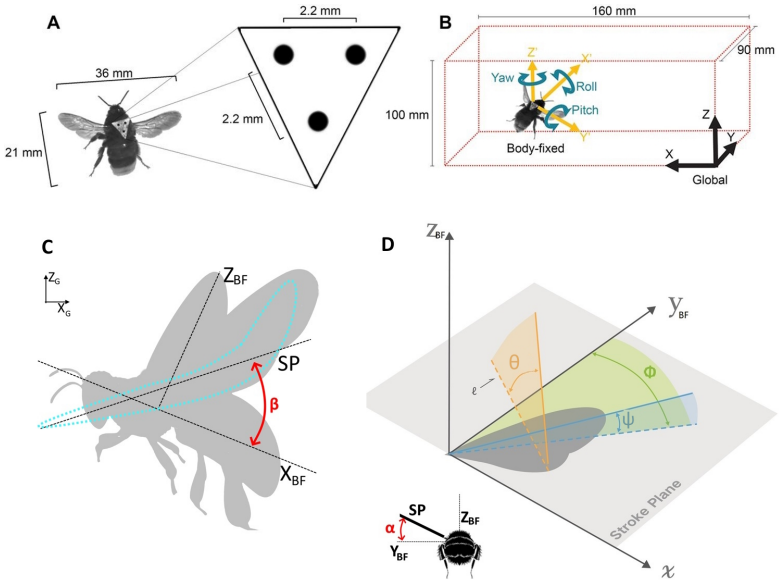


Figure 3: (A) The configuration of the trackable markers attached to the thoraxes of bees. (B) The global and body-fixed coordinate system axes used for defining bee motion. (C,D) The layout of the stroke plane relative to the global and body-fixed reference frames. (C) Shows how the wing moves relative to the bee body, with the blue dotted outline showing the path taken by the wing tip of a typical flapping stroke. The stroke plane angles relative to the body-fixed axes are illustrated for reference and were nearly horizontal. (D) Shows the wing angles for sweeping (ϕ), flapping (ψ), and feathering (θ) used to define the wing motion relative to the body-fixed coordinate system.

by manual verification of a subset of frames, ensuring reliable measurement of wing kinematics.

Three coordinate systems were defined. Firstly, the global coordinate system (XYZ_G) represents tracked bee coordinates within the flight tunnel (Fig. 3B). Second, the body-fixed coordinate system (XYZ_{BF}) identifies points relative to the body of the bee (Fig. 3B). This coordinate system is aligned with the southern point on the triangular marker, with the X_{BF} axis running longitudinally up the centre of the marker (and the bee body). The XYZ_{BF} is used for calculating body orientation changes and the stroke plane of each wing. Third, the stroke plane system is used for defining the angles of wing motion (Fig. 3C and D). For simplicity, the stroke plane coordinate system is considered the same as the body-fixed coordinate system but with offset angles between the marker (body-fixed coordinate system) and the stroke plane applied.

For wing motion, the flap, sweep and feather angles (ψ, ϕ, and θ) were defined in reference to the stroke plane (the mean plane through which the wing tip travels). Only wing sweep and flapping angle were analyzed in detail due to the high

http://www.imavs.org/

http://www.imavs.org/

sensitivity and complexities involved with minute changes in the feathering angle. The steady flight stroke plane of bees was defined through careful analysis of the wing motion during the phase of flight where no disturbance was present. The offset between the plane of the markers and the bee stroke planes were identified to ensure that the wing kinematics were represented in the relevant frame of reference.

The stroke cycle typically begins and ends at the zero stroke sweep angle, which is parallel to the Y_{BF} axis (with offset applied to the stroke plane) (Fig. 3D). However, due to the large variations in translational and rotational wing motions induced by the gusts in this study, the wing sweep angle varied significantly during the disturbance phases and independently between each wing. Consequently, it could not reliably define the start and end of the strokes. To uniformly identify the stroke phase during flight for both wings, the peak flapping angle (ψ_{Max}), which was more consistent across conditions and still coupled to wing sweep, was used as the reference point for the start of the stroke, with the timing of sweeping and feathering angles referenced to this peak. Wing flex deformations, due to both gust impact and wing compliance, were greatest at the peak angular amplitudes of the strokes. These could have introduced some error into the estimation of wing angles, but were considered adequately accounted for due to the process of averaging over multiple strokes in the results. Supporting information on the setup and related experimental procedures can be found in [21].

3 RESULTS

3.1 *Wing Motion In Unperturbed Flight*

Wing flapping angles in steady flight were analyzed to establish a baseline for comparison with gust-induced changes. The black curve in Fig. 4 represents the mean flap and sweep angles across a complete stroke, averaged from 108 wing stroke curves under steady conditions. The sweep angle had a stroke amplitude of $120.4° \pm 12.2°$, and the flap angle had an amplitude of $76.4° \pm 9.50°$ (Fig. 4), reflecting some variation due to mid-flight corrective behaviours. These findings are consistent with previous studies, such as in [22], which reported a maximum stroke amplitude of 115° to 125° in *Bombus terrestris*.

3.2 *Wing Motion Adaptations in Different Gust Conditions*

To capture the key changes in wing kinematics, we divided the flight into three distinct phases. The first phase, termed the steady phase, represents the period before the bees encountered the gust. The wing kinematics observed during this phase, as illustrated in Fig. 4, are included in Fig. 5 for direct comparison with the kinematic changes in the following phases. The second phase, termed the impulsive phase, occurred as the bees entered the gust but had not yet initiated any active responses. During this phase, all behaviours were passive or involuntary. The third phase, the recovery phase, began when the bees initiated counteractive manoeuvres in response to the gust. The wing kinematics during this phase represent the average stroke patterns as the bees attempted to return to their original flight path.

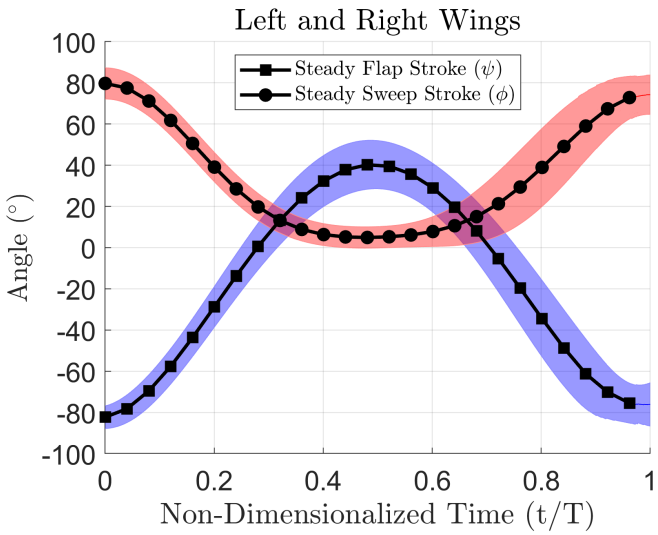


Figure 4: Mean stroke sweep angles (ϕ) and stroke flapping angles (ψ) (n = 108) plotted against non-dimensionalized stroke time (t/T). The start of the x-axis (t/T = 0) is the peak of the flap angle. Data from both the left and right wings were used to calculate the mean curves. The blue and red shaded regions indicate the standard deviation of the curves.

The bee body trajectory responses to gusts were published in a related study [19]. In the previous study, it was found that in flights through any gust direction, an impulsive orientation perturbation was generated by the gust in the direction the gust was pointing. For example, a sideward gust causes an impulsive negative yaw and a negative roll motion. When the bee engages in recovery, manoeuvres are undertaken against these perturbations. These related findings are summarised in Table 2.

Table 2: Summary of body orientation perturbations in gusts. The referenced perturbation directions are listed in Table 1.

Gust Condition	Perturbation	Recovery Manoeuvre
Sideward	-Yaw, -Roll	+Yaw, +Roll
Upward	+Pitch	-Pitch
Downward	-Pitch	+Pitch

3.2.1 Sideward Gusts

In sideward gusts, as bees penetrated the gust region, the stroke sweep angular range decreased to $55.1° \pm 15.5°$ for the wing closer to the gust (right wing), with the mean base value ($-79.5°$) aligning with the approximate minimum of the sweep range observed during steady flight ($-80.7°$, Fig.

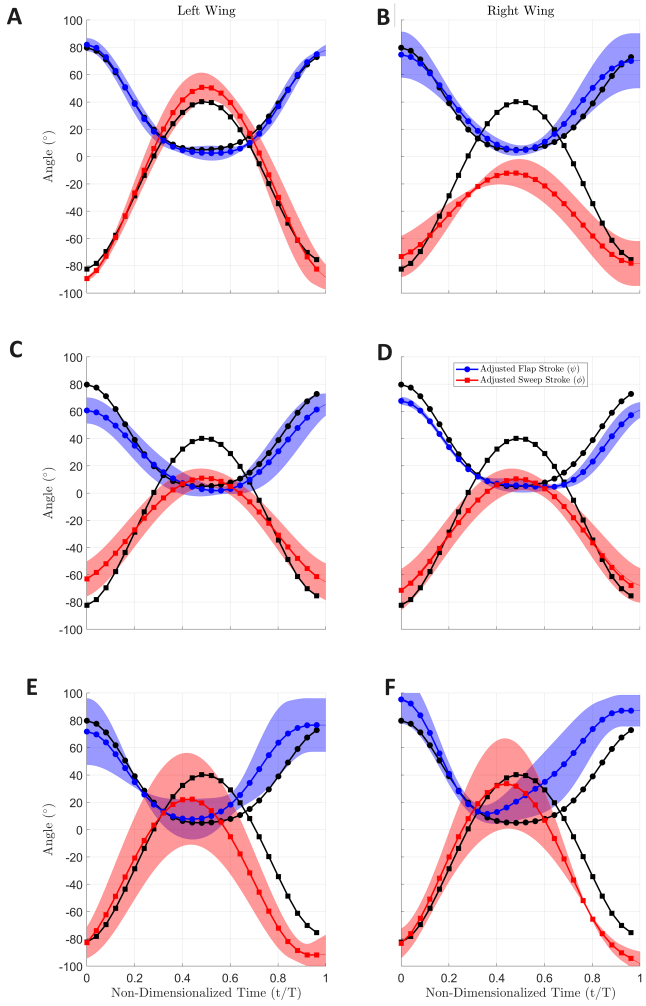


Figure 5: Mean stroke sweep and flap angles in each of the three gust orientations plotted against non-dimensionalised stroke time (t/T). (A) and (B) show the left and right wings in the sideward gust, (C) and (D) present the data in the upward gust, and (E) and (F) show the data in the downward gust. The blue and red shaded regions indicate the standard deviation of the curves. For reference, the mean flap and sweep curves for the steady case are shown in black.

4A). This posterior shift in the range of motion caused the right wing to remain behind the centre of gravity (CG), reducing the velocity and forces acting on the wing while maintaining a flapping frequency similar to that in steady flight. On the opposite side (the left wing), the stroke sweep angle increased to $130° \pm 10.3°$, extending beyond the full range of motion typical for these bees, as observed in the steady flight analysis. Despite these changes, the flap angle remained consistent for both wings, with similar angular ranges, reference positions, and timing, indicating no significant deviation from the steady flight condition (Fig. 5A).

3.2.2 Upward Gusts

In response to upward gusts, the stroke sweep angle of both wings shifted posteriorly to a reduced range, with the minimum point at $-64°$, closer to the minimum sweep observed during steady flight ($-80.7°$). Despite this shift, flapping frequency and phase timing remained consistent with steady flight, resulting in lower velocities over the wings (reciprocating over a shorter angular range). The forward sweep angle peak decreased to $8°$, confining most of the stroke range behind the CG at the onset of recovery. This adjustment effectively reduced the total stroke range to $72° \pm 11.6°$, nearly half of that observed during steady flight ($120.4° \pm 12.2°$). While the wing sweep was confined to this reduced posterior range, the flapping angles remained relatively consistent throughout the stroke cycle, peaking at $62.8° \pm 6.9°$, approximately $15°$ lower than in steady flight ($76.4° \pm 9.5°$). Shifting the wings posteriorly relative to the bee's body while maintaining the angular flapping range likely enhanced stability by keeping the wings behind the CG, optimizing the reception of gust forces and increasing the distance between the gust-induced drag and the CG, contributing to a more stable, bottom-heavy system.

3.2.3 Downward Gusts

In response to downward gusts, wing motions closely resembled those observed in steady flight, with wings sweeping across the full angular range and occasionally extending beyond it ($118° \pm 32.1°$). Both wings exhibited a 20% increase in stroke frequency, resulting in higher flow velocities over the wings as they reciprocated over a larger angular range. This increase in frequency was accompanied by greater variability in the strokes, with sporadic shifts in the anterior peak of one wing, leading to variations in the mean sweep and flap peaks ($\overline{\sigma_\psi} = 14.5°$, $\overline{\sigma_\phi} = 20.0°$ across both wings). These inconsistencies were likely associated with increased body roll and yaw oscillations during the perturbation. Additionally, the flapping angles were, on average, $12.7°$ higher than in the steady case across the stroke. The substantial increase in frequency and the tendency to sweep over a greater angular range likely represented an adaptive response by the bees to enhance flow velocity over the wings in reaction to the downward gust.

4 Inferred Dynamics from Kinematic Adjustments

In the experiments, bumblebees showed agility in their wing movements when responding to gust disturbances of different directions, dynamically articulating the wing towards unique configurations in each of the disturbance cases. Depending on the gust's direction, bees adjusted the mean position, range and frequency of their wing positioning relative to the body (through sweep and flap angles). Across all gust

http://www.imavs.org/

disturbances, sweep angle, or the anteroposterior configuration of the wing relative to the body was the main driver determining how bee attitudes changed in the gust. Variations in flapping angles were minimal compared to changes in sweep angles. While sweep and flap angles maintained coordinated adjustments throughout all responses to the gusts, changes in sweep angles did not result in corresponding changes in flapping angles within the same stroke, mostly maintaining its steady-flight articulation range across strokes.

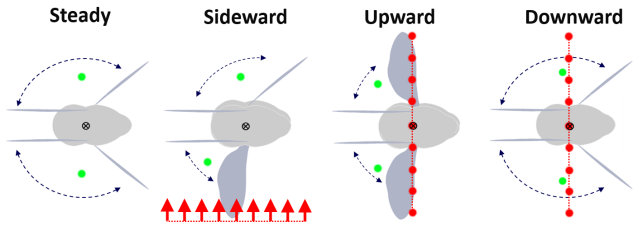


Figure 6: Wing kinematics in response to different gust directions. The diagrams categorise the wing kinematic configurations observed in each gust test case. Red arrows indicate the direction of the gust, while curved blue arrows represent the angular range of wing motion. The AC is marked by a green dot, and the CG is denoted by a central dark dot. Note that initial impulsive perturbations caused by the gusts are not depicted; the bee's position would actually be slightly displaced in the direction of the gust (red arrows) at the onset of these perturbations.

The main wing adaptive configurations in each of the three different gust directions are displayed in Fig. 6. In each case, the configuration generally appeared within 2-4 wing beats after experiencing the gust, and only lasted 1-2 wing beats before returning to a more steady stroke configuration (more similar to steady). This suggests that these brief, transient changes in the wing kinematics are critical in counteracting and recovering from gust-induced perturbations.

By considering the CG of bees to lie at the geometric centre of bees looking down (Fig. 6), and assuming aerodynamic forces act at an aerodynamic centre (AC) located near the wing's centre of area, the basic dynamics of observed wing configuration changes (Fig. 5) can be inferred. For sideward gusts, the wing closer to the gust shifted into a reduced sweep range. This effectively put the wing in a more posterior and dorsally raised position relative to the body for a greater proportion of the stroke. Using the illustration in Fig. 6 for reference, it can be seen that such a change would shift net wing forces further behind the CG (compared to the steady wing kinematics case), generating a restorative yaw moment toward the gust. Additionally, the reduced velocity on the wing closer to the gust results in an imbalance of upward force between the two wings, leading to a counteractive roll manoeuvre towards the gust. These roll and yaw corrections directly counteract the initial perturbation caused by the gust (Table 2) and facilitate a passive (utilising on forces from the gust) mechanism for handling sideward disturbances. Previous studies, such as [18], suggest that bees tend to prefer frontal gusts; hence, it is plausible that bees initially correct perturbations towards the gust and, if the gust persists (as is often the case in natural environments), overcorrect to face into the gust, a behaviour observed in some of the slower bees in the experiments.

In response to upward gusts, both wings shifted into a sweep range configuration confined to the posterior region. Similar to the sideward gust scenario, but involving both wings, this adjustment positioned the wings more posteriorly and dorsally relative to the body for a greater portion of the stroke. Although the reduced wing velocity would typically result in lower force generation, the posterior positioning of both wings behind the CG is conducive to generating a counteractive pitch-down torque. The upward force from the gust flow enhances the wing-generated forces in the same direction, behind the CG, thereby amplifying the pitch-down moment and countering the nose-up perturbation induced by the gust (Table 2). By maintaining both wings in the posterior position behind the CG, the bees appear to optimize stability, effectively utilizing their wing position to harness the gust flow and augment the counteractive forces.

In the case of downward gusts, which induced an initial impulsive nose-down perturbation, it might be expected that the bees would shift their wing sweep range anteriorly to generate a pitch-up response, similar to the strategy employed in response to upward gusts. However, the bees instead increased their flapping frequency while maintaining a broad sweep range, likely to enhance lift and counteract the downward force. This adaptation appears to be an effort to sustain lift production and preserve flight stability under challenging conditions, particularly under the adverse conditions created by downward gusts, which are known to disrupt flows near flapping wings [18, 21]. The findings suggest that downward gusts pose a particularly difficult scenario for flapping wing kinematics, as maintaining sufficient lift while counteracting the downward force necessitates more substantial adjustments in wing kinematics. This highlights a potential trade-off between generating the necessary upward forces and maintaining manoeuvrability during the disturbance, indicating that flapping wings may face inherent limitations in simultaneously achieving both objectives.

In two of the three gust orientations, one or both wings shifted their sweep range posteriorly, moving toward the lower half of the sweep spectrum and significantly reducing the median sweep angle. This shift likely enhances the stability of the bee by repositioning the aerodynamic forces relative to the CG, making the bee's response to disturbances more effective. this simple change for each wing can immediately provide stability enhancing effects, making it better for flight.

http://www.imavs.org/

http://www.imavs.org/

5 CONCLUSION

This study explored the adaptive wing kinematics of bumblebees under gust disturbances. The primary adaptive strategy observed was the posterior shift in wing sweep angles, which varied with gust direction. In sideward gusts, this shift in the wing closest to the gust enhanced lateral stability, while upward gusts induced a coordinated posterior shift in both wings to counteract pitch-up manoeuvres. For downward gusts, bees increased flapping frequency to maintain lift, though this may have limited their ability to counteract pitch-down perturbations.

These findings demonstrate the sophisticated control mechanisms employed by bumblebees to maintain stability in turbulent conditions. The posterior wing shift, a defining kinematic adjustment seen in the results, could enhance flapping manoeuvrability by dynamically adjusting wings with a similar shift during disturbances. Incorporating mechanisms to modulate wing position or balance force generation between wings could greatly improve MAV control and resilience in turbulent environments.

While this paper offers a qualitative analysis, further detailed statistical analyses will be presented in a forthcoming full paper, which will delve into the statistical significance of these findings and their implications for MAV design.

REFERENCES

[1] C. P. Ellington, C. van den Berg, A. P. Willmott, and A. L. R. Thomas. Leading-edge vortices in insect flight. *Nature*, 384:626–630, December 1996.

[2] Michael H Dickinson, Fritz-Olaf Lehmann, and Sanjay P Sane. Wing rotation and the aerodynamic basis of insect flight. *Science*, 284(5422):1954–1960, 1999.

[3] Sanjay P Sane. The aerodynamics of insect flight. *Journal of Experimental Biology*, 206(23):4191–4208, 2003.

[4] S. P. Sane and M. H. Dickinson. The control of flight force by a flapping wing: Lift and drag production. *Journal of Experimental Biology*, 204:2607–2626, 2001.

[5] Xinyan Deng, Luca Schenato, and S Shankar Sastry. Flapping flight for biomimetic robotic insects: Part ii-flight control design. *IEEE Transactions on Robotics*, 22(4):789–803, 2006.

[6] Bo Cheng and Xinyan Deng. Translational and rotational damping of flapping flight and its dynamics and stability at hovering. *IEEE Transactions on Robotics*, 27(5):849–864, 2011.

[7] L. Ristroph, G. J. Berman, A. J. Bergou, Z. J. Wang, and I. Cohen. Automated hull reconstruction motion tracking (hrmt) applied to sideways maneuvers of free-flying insects. *Journal of Experimental Biology*, 216:457–469, 2013.

[8] Robin J Wootton. Functional morphology of insect wings. *Annual review of entomology*, 37(1):113–140, 1992.

[9] James R Usherwood and Charles P Ellington. The aerodynamics of revolving wings i. model hawkmoth wings. *Journal of Experimental Biology*, 205(11):1547–1564, 2002.

[10] Liang Zhao, Qingfeng Huang, Xinyan Deng, and Sanjay P Sane. Aerodynamic effects of flexibility in flapping wings. *Journal of the Royal Society Interface*, 7(44):485–497, 2010.

[11] T Nakata, H Liu, Y Tanaka, N Nishihashi, X Wang, and A Sato. Aerodynamics of a bio-inspired flexible flapping-wing micro air vehicle. *Bioinspiration & Biomimetics*, 6(4):045002, 2011.

[12] S. Watkins, J. Millbank, and B.Loxton. Atmospheric winds and their effects on micro air vehicles. *AIAA Journal of Aircraft*, 44, 01 2006.

[13] Simon Watkins, Mujahid Abdulrahim, and Mark Shortis. Mitigating the effects of atmospheric turbulence: Towards more useful micro air vehicles. *Defense Technical Information Center*, page 27, 07 2010.

[14] A Fisher. *The effect of freestream turbulence on fixed and flapping micro air vehicle wings*. PhD thesis, RMIT University, 2013.

[15] Victor Manuel Ortega-Jimenez, Jeremy S. M. Greeter, Rajat Mittal, and Tyson L. Hedrick. Hawkmoth flight stability in turbulent vortex streets. *Journal of Experimental Biology*, 216(24):4567–4579, 2013.

[16] Sridhar Ravi, James Crall, Lucas McNeilly, Susie Gagliardi, Andrew Biewener, and Stacey Combes. Hummingbird flight stability and control in freestream turbulent winds. *The Journal of Experimental Biology*, 218, 03 2015.

[17] Alex Fisher, Sridhar Ravi, Simon Watkins, Jon Watmuff, Chun Wang, Hao Liu, and Phred Petersen. The gust-mitigating potential of flapping wings. *Bioinspiration and Biomimetics*, 11(4):46010–46010, 2016.

[18] Kamal Viswanath and Danesh K. Tafti. Effect of frontal gusts on forward flapping flight. *AIAA Journal*, 48(9):2049–2062, 2010.

[19] Timothy Jakobi, Dmitry Kolomenskiy, Teruaki Ikeda, Simon Watkins, Alex Fisher, Hao Liu, and Sridhar Ravi. Bees with attitude: the effects of directed gusts on flight trajectories. *Biology Open*, 7(10):bio034074, 2018.

[20] Tyson L Hedrick. Software techniques for two- and three-dimensional kinematic measurements of biological and biomimetic systems. *Bioinspiration & Biomimetics*, 3(3):034001, 2008.

[21] Timothy Richard JAKOBI. *Strategies exhibited by bumblebees when challenged with aerodynamic and obstacle disturbances*. PhD thesis, RMIT University, 2023.

[22] R Dudley and Charles P Ellington. Mechanics of forward flight in bumblebees: I. kinematics and morphology. *Journal of Experimental Biology*, 148(1):19–52, 1990.

http://www.imavs.org/

REGULATORY COMPLIANT UAS NAVIGATION BASED ON A BITMAP GRID LOCAL PATH PLANNING ALGORITHM

A. Pablo De Porcellinis*, B. Jesús Villadangos, C. Daniel Aláez, and D. Manuel Prieto
Public University of Navarre, Pamplona, Spain

http://www.imavs.org/

ABSTRACT

Infrastructure inspection nowadays is a mandatory task to maintain the life quality of the society. As these tasks have to be performed safely, ensuring that waypoints satisfy the most strict current Unmanned Aircraft Systems (UAS) regulations like not put in risk (overfly) items as persons, animals, etc., this work proposes an algorithm for UAS navigation during the power-line inspection that works autonomously. The proposed algorithm determines all along the flight path the most suitable regulatory compliant waypoints (RCWPs) to satisfy international regulations during the whole flight. This solution only requires knowing the exact position of the UAS during the flight and uses an artificial vision system for item identification in the environment along the route.

1 INTRODUCTION

From the beginning of modern civilizations, the use of different kinds of infrastructures has made easier the life of humans. Therefore, it is possible to establish a relationship between the life quality of the different countries nowadays and the quality of their infrastructures. To maintain the quality of these infrastructures it is mandatory to apply to them a maintenance. In the case of Spain for example, it is required by law to inspect the whole power-line infrastructure every three years.

There are several methods that have been applied to perform these inspections, but in the last decade the use of mobile robots and more specifically UAS, commonly known as drones, have become more common. The world of drones has a strong regulation and depending on the characteristics of the intended operation, a different risk level is assigned [1].

The infrastructures can be located in different scenarios that imply an increase in the level of risk, for example when they are close to urban areas or in contrast in very remote areas where the previous information about the operational scenario conditions is limited.

*Email address(es): deporcellinis.150939@e.unavarra.es

To reduce the level of risk of the operation, it is necessary to equip the aircraft with systems that improve its assurance and integrity levels and allow it to perform such inspections.

The level of risk assigned to the operations is mainly divided into ground and risk levels [2]. There are several items that are present in the areas of real inspections and need to be faced to overcome this risk. Some examples of these real-time problems that can be present in the inspection area are the wrong exact location of the power-line assets, uninvolved persons, moving vehicles, or even groups of grazing animals. The problem with these items is that it is not possible to predict them in advance, as they are real-time problems.

Although these elements are not considered obstacles, since they do not endanger the integrity of the aircraft, they should not be flown over to comply with the regulation. Therefore, it is necessary to come up with some solution to tackle the problem of these unforeseen actors and to allow the dynamic change of aircraft trajectory. More precisely, it is required a UAS navigation system that allows real-time and autonomous planning of the aircraft flight to ensure a safety flight that fulfills the requirements imposed by the regulator for the authorized inspection flights.

1.1 State of Art

In the literature, there is a wide variety of UAS navigation techniques that are artificial vision-based. In [3], they establish a taxonomy of the vision-based UAS navigation systems in three different groups depending on whether they are mapping-based methods for visual localization, object detection and avoidance approaches, or path planning-based approaches.

Regarding the path planning approaches, there are two different techniques used: Global Path Planning and Local Path Planning. The main difference between these two techniques is that Global Path Planning allows planning the path offline but requires the whole previous knowledge of the workspace and it is not possible to perform any change on the studied scenario while Local Path Planning allows to modify the path in a real-time basis, based on the characteristics of the environment [4].

In [5] a global path planning method using an A* algorithm is developed. This algorithm focuses on the mission planning using cost functions for computing the best route. This standard route algorithms are good for generating a min-

http://www.imavs.org/

imum cost route but are limited as they are based on a predefined cost function. The solution studied in [6] presents a different route planning approach being able to generate mission adaptable routes based on different constraints included in the cost function, such as minimum route length or maximum turning angle. Although this method is categorized as global path planning it is able to modify this constraints along the mission, getting closer to a more adaptive method. However, the mission parameters modification is not based on the visual detection of elements in real time.

Regarding global path methods that are capable of deal with obstacle detection, in [7] they introduce with a stepping neural network approach a way of producing fast and optimal motion plans for static environments. The problem of these global path methods is still that is not possible to modify the path in a real time basis, so it is necessary to completely know where the obstacles are placed prior to the UAS flight.

Looking for local path planning algorithms to find a real time path planning solution, there are several methods that are capable of avoid obstacles and find an optimized path at the same time. As an example of these methods, in [8] they purpose a combination of an Artificial Potential Field (APF) method and A-star fusion algorithm to avoid dynamic obstacles and find a shorter path simultaneously. The APF method use the gravitational field to control the flight direction of the UAV. The problem of this complex proposed algorithm is that is more suitable to fall into a local minima, and the path not being sufficiently smooth.

More recent studies as [9] are capable of produce a three dimensional (3D) vortex fields for path planning of different drones that works simultaneously in a swarm environment. This solution is capable of detecting moving obstacles and the sequential cooperation will allow to reduce the local minimum problem. For the proposed use case in this article, the cooperation is at the same time a problem, as it is not efficient to make a swarm flight to detect obstacles and find a local path. In addition, the detected obstacles are physical elements that if not avoided, will cause an accident of the aircraft.

Reference	Path Planning Type	Obstacles / Items Detection	Main Target	Safety Oriented
[5]	Global	No.	Cost Function. Distance Optimization	No
[6]	Global	No.	Mission adaptable routes (Route Length, Max. turning angle, ...)	No
[7]	Local	Yes. Static obstacles	Optimal motion plans with minimal performance loss	No
[8]	Local	Yes. Moving obstacles	Obstacle Avoidance and route length	No
[9]	Local	Yes. Moving obstacles	Path Oscillations and distance Travelled	No
RCLPP	Local	Yes. Moving Items	Safety Compliance. Regulatory Basis.	Yes

Table 1: Path Planning Algorithms Comparison

In this paper it is proposed a new regulatory compliant local path planning algorithm (RCLPP) which evaluates dynamically the UAS path. The path calculated is composed by Regulatory Compliant Waypoints (RCWPs).Each one ensures items location and local path planning strategies to find a feasible solution from the definition of a security region about those items.

The use of this classical definition of safety region has been previously used for a global path planning algorithms but has not deeply exploited for local path planning methods. For the proposed use case the main objective is to face the problem of not flying through a moving item to be compliant with the regulation limitations, so therefore is not necessary to apply a cost function [10] to decide the more cost effective solution based on desired parameters, as the first viable solution reached by the algorithm is the one that will be selected.

The main difference between the proposed method and the ones presented on the state of the art is that is the only safety oriented solution with a regulatory basis, to comply with the constraints of the authorized inspection flights.

2 REAL TIME UAS NAVIGATION ALGORITHM

This section is divided into two main parts. In the first part, the real-time UAS navigation algorithm is explained, where a previously defined flight plan is complemented with the local path planning solution for those regions where regulatory problems have to be faced.

In the second part, this local path planning solution is explained in detail, highlighting one by one the functions used to find the regulatory compliant path.

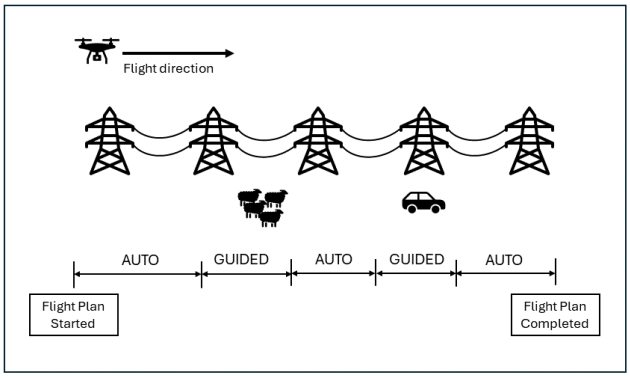


Figure 1: Expample of the proposed solution

In Figure 1, it is represented graphically the operating concept of the proposed solution. The idea is to complete the power-line inspection in an automatic way following a Flight Plan. The special characteristic of this Flight Plan is that is dynamic and if some regulatory challenging items appear in the expected trajectory of the UAS, it is capable to perform a Flight Plan reconfiguration. As can be seen, when the items

are detected, the flight mode has changed between "AUTO" and "GUIDED", to be able to adopt this real-time Flight Plan reconfiguration.

The difference between the used flight modes for this solution is that in the "AUTO" mode, the WPs are automatically introduced and followed based on a predefined route while in the "GUIDED" the UAS processes only one WP at the same time, so it is necessary to set the next target point each time the previous one is reached. Therefore, each of them is suitable for specific flight conditions.

```
1   FlightPlan <- read_FlightPlan()
2   flight_start(ARM, AUTO)
3   while (FlightPlan is not completed)
4         RCWPs <- emptyset
5         while (items_detected(onBoard_camera) == false)
6               NextWP <- get_nextWP(currentGPS, FlightPlan)
7               RemainingFlightPlan <- update(FlightPlan)
8               Flight_mode = AUTO
9         while (items_detected(onBoard_camera) == true)
10              ItemsLocation<- calculate_ItemsLocation()
11              RCWPs <- LPP(itemsLocation, currentUAVPos, last(FlightPlan))
12              If RCWPs <> emptyset
13                    NextWP <- second(RCWPs)
14                    Flight_mode = GUIDED
15                    RemainingFlightPlan <- update(FlightPlan)
16              Else
17                    Flight_mode = LAND
```

Figure 2: Real Time UAS Navigation Algorithm

The Flight Plan for the mission was previously designed based on the position of the power-line assets. This Flight Plan is designed using a map-based solution where additional aspects such as the communication network coverage, external services coverage (GNSS services, U-Space, etc), or already obstacle locations play an important role in the definition of the Flight Plan WPs.

The real-time UAS navigation algorithm is described in 2. In the first step of the proposed method, the $read\_FlightPlan()$ function reads the predefined route WPs and transfers it to the command unit of the aircraft, providing it with what is needed to develop the flight in an automatic way.

The next step is to start the flight, providing the command for arming the aircraft and setting the "AUTO" flight mode, with which the UAS is capable of autonomously following the predefined Flight Plan.

While the flight plan is not completed, the proposed solution will continuously check in which of the possible situations the aircraft is flying in. The system responsible for providing the input for the selection of the method is the RGB camera on-boarded in the UAS. This forward-looking optical sensor captures in real time the flight environment information to search for any item that should not be flown over to comply with the regulation.

To localize the items, the methodology for 2D object detection and tracking proposed in [11] is used. This methodology is based on assigning a rectangular Region of Interest (ROI) to detected items by a camera over the course of successive video frames. This detection method combines a complete Robot Operating System (ROS), a fast object detector (You Only Look Once (YOLO) series [12]) and a visual object tracking method.

As the YOLO network need to be trained to detect with accuracy the desired type of elements, in this case the training was performed to recognise groups of people and grazing animals. The reason for selecting those type of elements principally is because they are identified as the most common on-ground regulatory barrier elements [1]. However, as the YOLO has the capacity of being trained, the proposed model allows, if required, to add new elements to be detected in the planning phase, depending on the characteristics of the scenario and on user needs.

While the onboard camera does not detect any potential item, the proposed UAS navigation method will command to the aircraft to continue flying in AUTO mode. This flight mode is based on a global path planning method, in which all the proposed waypoints are fixed prior to the start of the flight. For this flight mode, the system will continuously check in each iteration $get\_nextWP()$ function which is the next pre-planned WP to be reached, and commands automatically to the UAS to reach this position. To search for this WP the function uses as input the GNSS position of the UAS in each captured instant and compares it with the pre-planned flight plan, to select the closest one to this position.

Inside the same status loop, the $update()$ function is used to update the status of the pre-planned Flight Plan. It has two main functionalities:

- Update the Flight Plan to consider in the next iteration only the remaining WP of the pre-planned route that is not left behind yet.
- Set the Flight Plan as "completed" once it is finished, to end the loop and finish the flight.

If a potential item is detected, i.e. $items\_detected()$ function is true, the proposed method will swap to the second status loop described in Figure 2 and will remain inside this loop until the onboard camera does not detect any potential item in the next iteration. Inside this loop, the proposed UAS navigation method will command to the aircraft to fly in "GUIDED" mode.

The first step of this status loop is to transform the item information captured by the camera into a location for the item. To perform this task, the methodology described in [11] is used.

This methodology first identifies the position of the bounding box relative to the center of the camera. Once it

[1] The grazing animals training was used also by UPNADrone team to fulfil the requirements for the IMAV 2024 outdoor competition

http://www.imavs.org/

http://www.imavs.org/

is identified, it is possible to obtain the distance between the item and the camera using the pitch angle θ of the camera, the vertical angle between the rays that project to the camera focal center and to the middle bottom of the boundary box and the height of the aircraft h. After this distance computation, a transformation between the camera reference frame and the North-East-Down (NED) coordinate system [13] is used (in the camera reference frame the X axis is pointing in the forward direction of the camera plane and the Y axis is pointing right in the camera plane). This transformation is performed to have the position of the items in the common reference frame used for navigation.

Once the position of the items is known, the status loop will arrive at the $LPP()$ function, where the Local Path Planning Algorithm described in the following section is used to fulfill the objective of finding a new route based on RCWP.

Once the status loop exits the $LPP()$ function there are two possibilities:

- The function gives as output a RCWPs empty set, which means that is not possible to find a regulatory compliant path, so the $Flight\_mode$ changes to "LAND" to perform a safety maneuver.

- The function gives as output a RCWPs matrix with the new regulatory compliant track to follow. As the $Flight\_mode$ is "GUIDED", the UAS will take the first element of the set as the next location to be reached.

2.1 *Local Path Planning (LPP) Algorithm for Regulatory Compliant (RC) Waypoints Selection*

In this subsection, the Local Path Planning Algorithm (LPP) is explained in detail, to understand how the regulatory compliant path is computed based on the detection of real-time identification of non-compliant items. This algorithm is described in Figure 3.

```
LPP(itemsLocation, currentUAVPos, last(FlightPlan))
    Grid(1:N, 1:N) = 1
    Prohibited, start, end <-insert_items(itemsLocation, Grid, currentUAVPos,...
    ...last(FlightPlan))
    Updated_Grid <- set_RC_safetyareas(Prohibited, Grid, RC_safety_factor)
    Final_Grid <- safety_areas_environment(Updated_Grid. RC_safety_factor)
    InitialPath <- get_initial_path(start, end, Final_Grid)
    FinalPath <- get_optimized_path(InitialWPs)
    RCWPs <- points_to_RCWP(FinalPath)
```

Figure 3: LPP Algorithm for RCWPs selection

This algorithm works with a grid representation with dimensions NxN. Each of the grid points represents a combination of latitude and longitude GNSS coordinates.

The N parameter can be selected depending on different variables of the problem, such as how reactive is the UAS being used or the separation between the GNSS coordinates translated into the grid, as N is directly related to the time to reach the next contiguous point.

The algorithm converts the grid representation into a bitmap, depending if the coordinate can be reached by the UAS or not. Being the grid a matrix $G\{NxN\}$ and $x, y \in N$ the possible coordinates inside the matrix, G(x,y) = 1 is a reachable position, and G(x,y) = 0 is blocked by an item position. Each time that the $LPP()$ function runs, it is assigned to each grid point an initial value of 1.

Figures 4, 5, 6, 7 and 8 will represent different stages of the solution in the form of a bitmap grid. These figures are used simply as a graphical sample since the bitmap grid is not a metric setting.

In the first stage of the computation with the function $insert\_items()$, the algorithm reads the coordinates of the items seen by the camera and relates them to their relative position in the grid. In each of the grid points where an item has been placed, the algorithm automatically changes its value to 0, meaning that at this instant, this position is blocked and can not be selected as a reachable point by the aircraft. This function additionally reads the current UAS position in that instant and the next WP from the Flight Plan and assigns them to the variables $Startpoint$ and $Endpoint$ respectively.

As it has been mentioned before, the main objective of this proposed solution apart from completing the power-line inspection correctly is to comply with the regulation, guaranteeing the safety of the flight and the surroundings. This is exactly what is done inside the function $set\_RC\_safety\_areas()$, applying a safety margin around the position of the detected items. For the results presented in this article, a safety margin of $R = 1$ is applied to make the solution simpler. This safety margin is translated into the bitmap grid, assigning to the neighbor cross points from the item position (i.e., grid points located at North, South, East, and West positions from the reference point) a new value of 0 for blocking it.

Once the items have been located and placed into the grid, it is important to analyze them together, to see if they are located on nearby grid points. If they are sufficiently close, the safety areas will block the path. Therefore, it is necessary to make these identified points not reachable. For the example representation it is considered again a safety margin of R = 1. Considering G(x,y) as the item position, if there is an item placed at G(x,y-2), it is necessary to assign to G(x-1,y-1) and G(x-1,y-2) a value of 0, to block this points to find a feasible path without crossing safety margins. The same logic applies to an item that is placed at G(x, y+2). This process is done inside the $safety\_areas\_environment()$ function.

After all the items and their surroundings are correctly identified in the bitmap grid, the algorithm looks for a feasible item avoidance path using the function $get\_initial\_path()$. In Figure 4 is represented this initial regulatory compliant path in a bitmap grid format. The variables $Start$ and End are used as the initial and final points of the computed initial

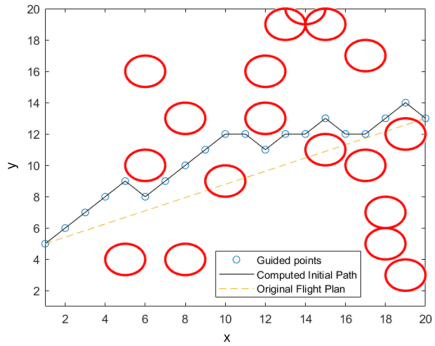


Figure 4: Initial Regulatory Compliant Path

path. There are three cases for the path computation:

1. No blocked space by the detected items and their safety margins (i.e. there are no 0s in the neighbor grid points): the algorithm will try to reach by each movement the y coordinate of the end reference point. In Figure 4 for example, it advances from $G(3,7)$ to $G(4,8)$, as the y coordinate of the final point is 13. If the y coordinate of the path point is equal to the y coordinate of the end reference point, the path will advance straight to the endpoint. In the reference example in Figure 4 this case is not represented, as there are always obstacles in the surroundings when the path reaches the y coordinate of the final point.
2. Partially Blocked space by the detected items and their safety margins (i.e. there are 0s in the neighbor grid points): the algorithm will select the closest free space in the next column of the bitmap grid. In Figure 4 for example, it advances from $G(5,9)$ to $G(6,8)$ to avoid the item placed in $G(6,10)$ and its safety margin.
3. Completely Blocked space by the detected items and their safety margins: the algorithm will finish the function automatically, returning $InitialPath$ as an empty-set.

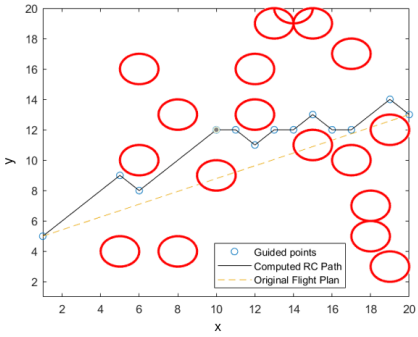


Figure 5: Final Regulatory Compliant path

Once the Initial Path computation is finished the algorithm uses the function $get\_optimized\_path()$ to refine the proposed path and eliminate all the nonrelevant points to achieve the Regulatory Compliant path. This process of path point number reduction is done to assure and facilitate the proper functioning of GUIDED MODE. This final Regulatory Compliant path is represented in Figure 5, again in a bitmap grid format.

As a final step, the $points\_to\_RCWP()$ function transforms the final optimized path grid points into GNSS coordinates, to be able to enter them as feasible Regulatory Compliance Waypoints (RCWPs) for the aircraft GUIDED MODE.

It is considered that the algorithm is capable of reacting to the changes in the environment at a prudent distance as in the represented example. In case the detected item performs a position change when the UAS is sufficiently close to it, there are some emergency procedures defined by the UAS operator to solve the situation safely and securely. These procedures are required by regulation to develop the intended inspection operations.

3 RESULTS

In this section, the results of the experiments performed in Matlab using the proposed method are represented. Firstly, the path planning reconfiguration is tested, to see if it is capable to react to the movement of an item.

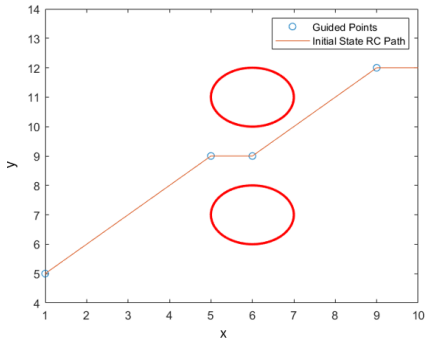


Figure 6: Moving Item Initial State Representation

Figure 6, 7, 8 represents in a bitmap grid format the different stages of a moving item detection and path reconfiguration.

In Figure 6 it is represented the initial situation after the items detection. The UAS is positioned at $G(1,5)$ and the items are initially detected at $G(6,7)$ and $G(6,11)$. As it is appreciable, there is an initial path planning for the guided mode based on this situation. Once the UAS advances one of the items changes its position. This movement is recognized by the camera and translated to the path-planning algorithm.

As it can be seen in Figure 7, the UAS has moved to position $G(1,7)$ (this position corresponds to first frame position $G(3,7)$ in Figure 6, as the UAS has advanced in the "path direction" following the proposed new route). The items are now positioned at $G(3,9)$ and $G(3,11)$, blocking the previous regulatory compliant path.

http://www.imavs.org/

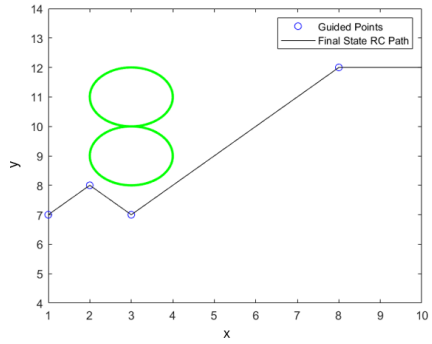


Figure 7: Moving Item Final State Representation

In Figure 8 is represented a combination of both instants, to validate that the algorithm is capable of dynamically changing at every instant, so it proposes new following guided mode points to avoid the items that change its position. As can be seen, the dashed black circles represent the movement followed by the item in previous states, while the green circles represent the safety margin of the items at the final state position. Similarly, the blue dashed line represents the initial state regulatory compliant path, while the red line represents the final state regulatory compliant path.

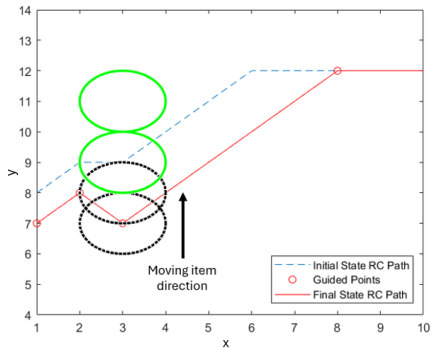


Figure 8: Path reconfiguration after an item movement

Secondly, the Local Path Planning Algorithm for Regulatory Compliant WP Selection was tested for different number of detected items. In total, 25 tests were performed for each of the values.

The Table 2 summarizes the results obtained. The parameter N is set as $N = 20$ for the experiments performed. As it is appreciable, if the number of detected items is smaller than the value of the characteristic dimension of the grid (N), no emergency landing is reported, as it is not probable that the path gets blocked. This is due to the fact that besides having few elements, they are placed randomly, so the probability of being placed surrounding the UAS position at any instant is almost null.

However, when the number of items is increased up to be equal to N, it is appreciable that even the low probability, an emergency landing due to a blocked path was reported. In addition, the number of necessary RCWPs to get the regulatory compliant path increased.

Number of detected items	Number of Guided Points	Emergency Landings
5	5	0
10	7	0
20	9-10	1
30	11-12	3

Table 2: LPP algorithm computation results

Finally, it is observed that when the number of detected items is greater than the characteristic grid dimension, the probability of an emergency landing due to a not feasible path increases as the representative space is saturated. Additionally, it is necessary that the UAS make movements in more than half of the given positions in the bitmap grid.

4 PUBLISHING POLICY

The IMAV proceedings will only be distributed in digital form. The submitting author is responsible for obtaining agreement of all his / her coauthors and any consent required from sponsors before submitting a paper. The authors are obliged to cite relevant prior work.

5 CONCLUSION

In order to solve the problem of infrastructure inspection the use of mobile robots has been used. There are several methods to automatically define the path that has to be followed by the robots, but none of them is capable of dealing with the regulatory barriers caused by the unexpected moving items. We propose a brand new real-time UAS navigation algorithm that is safety-oriented and combines local and global path planning to comply with the constraints imposed in real-time by the unexpected incoming moving items along the previously planned path. We perform a description of the proposed solution, based on a combination of two different flight modes and a regulatory compliant local path planning algorithm, capable of assigning a safety region around the items and modifying the previous planned path based on the unforeseen items. We perform simulations to evaluate the performance of the proposed method and these simulations reflect that is a feasible solution if the presence of items is reasonable to not saturate the sampling space. The next step is to apply it to a real scenario and improve the algorithm to include cost functions that select the best path based on desired parameters, such as less energy consumption, the shortest path, ... In this case, the solution presented has an on-ground focus so the camera objective is front-down oriented to detect the objects on the ground, but future experiments can include a up-front oriented camera to detect aerial items (always with a YOLO training with new item before) and add new use cases to the solution.

http://www.imavs.org/

REFERENCES

[1] EASA. Easy access rules for unmanned aircraft systems. https://www.easa.europa.eu/en/downloads/110913/en, 2024.

[2] JARUS. Jarus guidelines on specific operations risk assessment (sora). http://jarus-rpas.org/sites/jarus-rpas.org/files/jar_doc_06_jarus_sora_v2.0.pdf, 2019.

[3] Muhammad Yeasir Arafat, Muhammad Morshed Alam, and Sangman Moh. Vision-based navigation techniques for unmanned aerial vehicles: Review and challenges. *Drones*, 7(2), 2023.

[4] Pablo Marin-Plaza, Ahmed Hussein, David Martin, and Arturo de la Escalera. Global and local path planning study in a ros-based research platform for autonomous vehicles. *Journal of Advanced Transportation*, 2018:1–10, 2018.

[5] D.M. Rouse. Route planning using pattern classification and search techniques. In *Proceedings of the IEEE National Aerospace and Electronics Conference*, pages 2015–2020 vol.4, 1989.

[6] Robert J Szczerba, Peggy Galkowski, Ira S Glicktein, and Noah Ternullo. Robust algorithm for real-time route planning. *IEEE Transactions on aerospace and electronic systems*, 36(3):869–878, 2000.

[7] Mayur J Bency, Ahmed H Qureshi, and Michael C Yip. Neural path planning: Fixed time, near-optimal path generation via oracle imitation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3965–3972. IEEE, 2019.

[8] Chunyu Ju, Qinghua Luo, and Xiaozhen Yan. Path planning using artificial potential field method and a-star fusion algorithm. In *2020 global reliability and prognostics and health management (PHM-Shanghai)*, pages 1–7. IEEE, 2020.

[9] Rafael Monteiro Jorge Alves Souza, Gabriela Vieira Lima, Aniel Silva Morais, Luís Cláudio Oliveira-Lopes, Daniel Costa Ramos, and Fernando Lessa Tofoli. Modified artificial potential field for the path planning of aircraft swarms in three-dimensional environments. *Sensors*, 22(4):1558, 2022.

[10] Andreas Thoma, Karolin Thomessen, Alessandro Gardi, Alex Fisher, and Carsten Braun. Prioritising paths: An improved cost function for local path planning for uav in medical applications. *The Aeronautical Journal*, 127(1318):2125–2142, 2023.

[11] Daniel Aláez, Vasileios Mygdalis, Jesús Villadangos, and Ioannis Pitas. Real-time object geopositioning from monocular target detection/tracking for aerial cinematography. In *2023 IEEE 25th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE, 2023.

[12] Rachit Mayur Shah, B Sainath, and Akshansh Gupta. Comparative performance study of cnn-based algorithms and yolo. In *2022 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, pages 1–6. IEEE, 2022.

[13] Irene Santos and Anibal Ollero. Deteccion de obstaculos lidar mediante la tecnica de background. Master's thesis, Universidad de Sevilla, 2013.

http://www.imavs.org/

Robust Heading Estimation from Polarization Images by Deep Neural Networks

L. Yadala Chanchu[*], R.F.G. Zuidgeest[*], D.M. Stam[†], G.C.H.E de Croon[‡]

ABSTRACT

Heading estimation is vital for the autonomous flight of unmanned aerial vehicles. Magnetometers are typically used for this purpose, but they are not robust to electro-magnetic interferences. As a promising alternative, we investigate the insect-inspired solution of skylight polarization sensing. In particular, we develop a robust polarization compass for azimuth estimation. Two datasets are created - one based on a Mie scattering simulation, and one containing real-world pictures captured with a polarization camera under a variety of weather conditions. We employ the ResNet-18 model, which is trained and tested on both datasets separately. The trained model is robust to different weather conditions, and is able to directly analyze maps in the instrumental plane. The median error on the (mostly cloudy) real-world images of 4.30 degrees makes it a promising new method for the navigational toolkit of UAVs. We publish the real-world polarization dataset as open access data, in order to facilitate improvements by the community.

http://www.imavs.org/

1 INTRODUCTION

The rise of autonomous systems has brought with it the challenge of navigating effectively and robustly in the multitude of environments these systems can be used. As the demand for unmanned aerial vehicles (UAVs) continues to grow in a variety of sectors, challenges associated with reliable navigation under non-ideal conditions become more relevant. UAVs need to navigate effectively also under adverse environmental conditions which can include limited or blocked visibility and disrupted satellite signals.

Conventionally, drone navigation is done through the use of GPS, inertial measurement units (IMUs), and magnetometers [1], but these methods have significant shortcomings. GPS relies on clear signal communication to satellites, which is not possible in the case of large obstructions. On the other hand, IMUs will accumulate errors while the drone is flying [2]. Magnetometers are sensitive to external influences like electrical cables or large metal structures such as ships [3]. For this reason, skylight polarization is being heavily researched as an alternative navigation method.

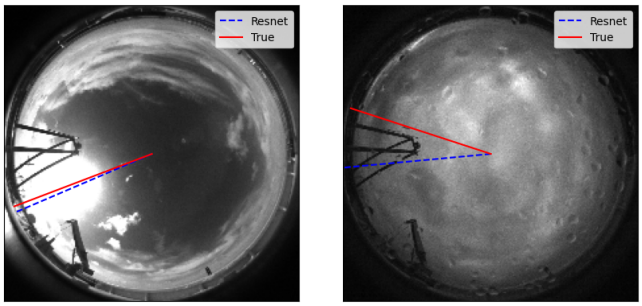


Figure 1: Sun direction estimation by a deep neural network based on polarization images. Red lines are ground-truth sun directions based on location and time. Blue dashed lines represent the predictions of the trained ResNet model. Predictions shown for (left) a clear sky and (right) an overcast sky.

Work on skylight polarization has mainly focused on azimuthal angle determination in clear sky conditions to test their efficacy. Traditional hand-made methods are often verified with captured clear sky images or simulations based on Rayleigh scattering. Newer models based on machine learning (ML) are predominantly validated on simulation datasets, which might not reflect the real world imperfections that occur - especially with the presence of heavily overcast skies. Related work has utilised many different approaches, from exploiting the anti-symmetry present [4] along the solar meridian in Angle of Polarisation (AOP) maps to ML-based techniques which experiment with different data label encodings [5]. However these methods focus on performance under ideal conditions, typically under near-ideal sky conditions or simulation data.

It has been observed that non-ideal environmental conditions like cloud cover significantly reduce the efficacy of present methods to determine heading [6]. In this study, the aim is to further investigate how a machine learning-based approach behaves under less-than-ideal sky conditions and compare performance when trained on simulated images.

2 BACKGROUND

Polarization preliminaries

Polarization cameras capture polarized light intensity in four different directions using a polarization array, shown in

[*]Faculties of EEMCS and ME at Delft University of Technology, equal contribution
[†]Leiden Observatory
[‡]Faculty of AE at Delft University of Technology, contact: g.c.h.e.decroon@tudelft.nl

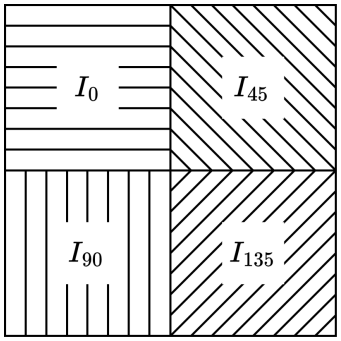


Figure 2: The polarization array of a single pixel.

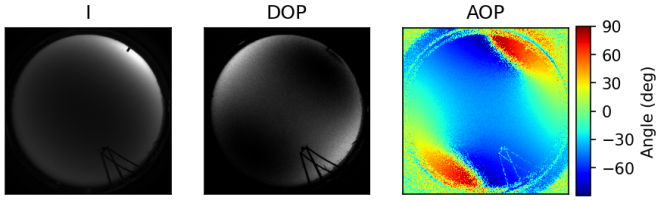


Figure 3: Intensity map with corresponding DOP and AOP.

figure 2. This polarization information is then translated to the Stokes parameters I, Q and U using equation 1. These values describe the polarization state of electromagnetic radiation [6]. There is a fourth parameter, V, describing circular polarization and is negligible for sunlight in the earth's atmosphere.

$$\begin{cases} I = \frac{1}{2}\left(I_0 + I_{45} + I_{90} + I_{135}\right) \\ Q = I_0 - I_{90} \\ U = I_{45} - I_{135} \end{cases} \tag{1}$$

With the Stokes parameters of all pixels, polarization maps can be created. Two common maps are degree of polarization (DOP) and angle of polarization (AOP) maps. Examples of these maps are shown in figure 3. The formula for calculating these maps is shown in equation 2.

$$\begin{cases} DOP = \frac{\sqrt{Q^2+U^2}}{I} \\ AOP = \frac{1}{2}\arctan\left(\frac{U}{Q}\right) \end{cases} \tag{2}$$

There are two reference frames that can be used for the angle of polarization (see figure 5): the instrumental frame (left, fig. 5) - representing the polarisation directly from the measured input and the global frame (right, fig. 5) - representing the polarisation angle with respect to the solar meridian. Conventionally, navigation algorithms rely on the use of an AOP map in the global reference plane. In this reference frame, the solar meridian is a straight line - the axis of central symmetry. This reference frame can be identified by the characteristic '8-shape' visible in the AOP map. AOP-based navigation is a popular choice, as the AOP map is more robust to cloud cover and other aerosols [7].

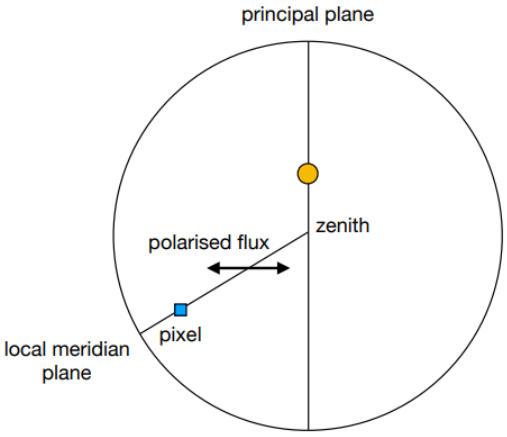


Figure 4: The 2 reference planes projected on a 2D plane, $\beta = 120$ deg (measured clockwise).

The problem arises in obtaining maps in the global reference plane (also known as the principal plane). To transition from the local (instrumental) plane, the stokes parameters are adjusted according to equation 3. A variable rotation Mueller matrix is applied to each pixel, with angle β measured clockwise in degrees from the pixel to the solar meridian [8]. Figure 4 schematically shows how the transformation is carried out. Then the AOP is updated with formula 2. The problem with this transformation, however, is that the location of the solar meridian must be known beforehand. This means that this approach is not usable for applications where the solar meridian is unknown to begin with.

$$\begin{bmatrix} I' \\ Q' \\ U' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(2\beta) & sin(2\beta) \\ 0 & -sin(2\beta) & cos(2\beta) \end{bmatrix} \begin{bmatrix} I \\ Q \\ U \end{bmatrix} \tag{3}$$

Non-ML Algorithms

Several hand-made approaches have been proposed to find the solar meridian in the global plane AOP map. A commonly used baseline is a least squares fit. One such method

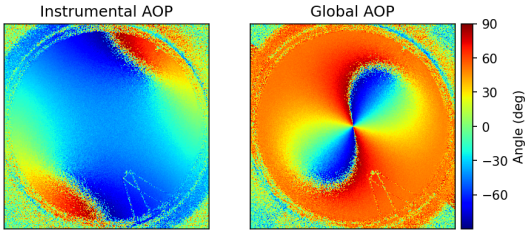


Figure 5: The two reference planes for AOP maps.

http://www.imavs.org/

http://www.imavs.org/

tries to find the optimal fit with a symmetry scanning approach of randomly sampled pixel pairs [7]. Other methods are based on fits with a moment of inertia approach [4] or comparing the rotation of pixel rings of sequential AOP maps [9].

DOP-based navigation do not rely on fitting a symmetry line. For example, SkyPole is a novel method that is able to find the celestial pole from a sequence of DOP measurements [10]. In contrast, we are interested in a method that maps a single image to the sun direction.

ML-Based Methods

ML models are able to learn what polarization features are useful for decoding orientation information.

One such approach is based on the intensity map, DOP and AOP [5]. These maps are fed to a fully connected (FC) network, which then outputs a heading estimation. Preprocessing of images with neural networks is also being researched. PCA-Net has showed to be useful in denoising real-life AOP maps [11].

3 METHODOLOGY

In this article, we investigate the use of deep neural networks for mapping sky polarization images to sun direction directly. In this section, we explain the neural network model and dataset.

3.1 ResNet Model

We utilized the ResNet-18 model, which has been pretrained on the ImageNet dataset, due to its capability to effectively learn and extract significant features from a variety of images automatically [12]. Polarization information from the sky can contain subtle patterns and gradients that are crucial for determining heading direction, and ResNet-18's deep architecture can effectively capture these features.

Special care must be taken in the way the azimuth angle is encoded. Previous research has shown that the best neural network performance is achieved when azimuth estimation is transformed to multi-class classification, as opposed to a regression task [5]. The proposed encoding method is known as exponential encoding, $N_e(k)$.

$$\begin{cases} N_e(k) = m^{|i|}, & 0 < m < 1 \\ i = k - \frac{\phi}{j} \end{cases}$$

Where $k \in \left[0, \frac{360}{j}\right]$ denotes the index of the output node, $0 < \phi < 360$ is the encoded angle in degrees, j is the resolution, and m is the constant determining the exponential decay. In essence, i describes how many indices a node k is located away from the encoded angle. Exponential encoding ensures that nodes coherently share the information of orientational mapping. Figure 6 shows an example of an exponential encoding probability density function with a resolution of $j = 0.1$ deg. For this resolution, the last fully connected layer of the ResNet model is altered to contain 3600 output nodes.

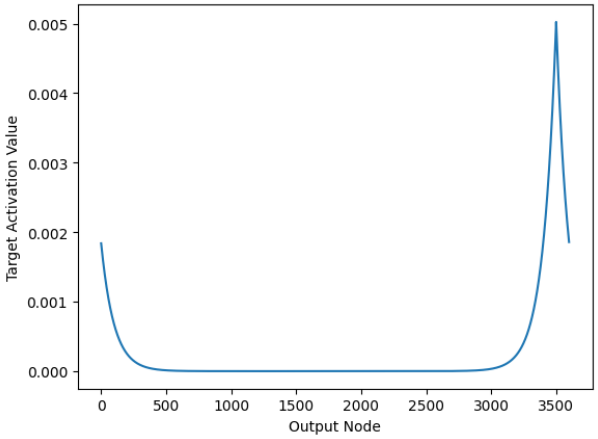


Figure 6: Exponential encoding ($m = 0.99$, $\phi = 350.0$) normalized with a Softmax

Note that by normalizing the exponential encoding to resemble a probability mass function (using a Softmax), the cross-entropy loss metric can be used during training.

3.2 Setup and Dataset

We used two different methods to gather the data required to train the model. The first method creates a synthetic dataset, based on a theoretical model for polarization. This model takes into account Mie scattering and is therefore more realistic than the standard Rayleigh model [13].

The simulation outputs Gaussian quadrature points for a map of the sky hemisphere. With the help of the Python OpenCV library, the I_0, I_{45}, I_{90}, I_{135} intensity maps are acquired and scaled to 224 x 224 pixels. To simulate the information loss that a cloudy atmosphere would introduce, it is possible to randomly obscure information in elliptical areas. This approach mimics the loss of information a cloud can introduce, and is used in the PSNS dataset [14]. Between 1 and 5 ellipsoidal clouds are randomly added to the images, with center pixel locations drawn from $\mathcal{N}(112, 28^2)$, axis lengths of $\mathcal{N}(224/5, (224/25)^2)$ pixels, and a random integer rotation of $\mathcal{U}(0, 90)$ degrees.

The second method captures real-world data with a physical setup, located on the rooftop of the faculty of aerospace engineering at TU Delft, in Delft, the Netherlands. Real-world pictures were captured with the Sony XCG-CP510 camera and the FE185C057HA-1 Fujifilm fisheye lens. The experimental setup is shown in figure 7. The images were collected from 8:00 until 19:00 with picture intervals of 10 minutes to get a varied dataset. At the time of writing, the setup has collected 2618 images over the course of 4 months, from January 2024 until April 2024*. The camera was regularly

*With some breaks between due to overheating and exposure issues

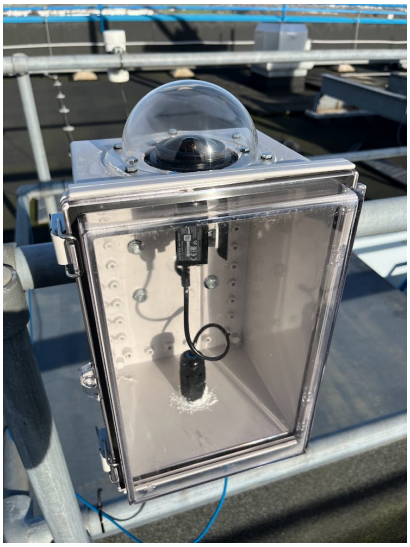


Figure 7: Set-up as mounted on the rooftop

http://www.imavs.org/

rotated during this period. More details about this dataset can be found in the Appendix.

Rotation of the camera is necessary, as there are fixed obstructions on the rooftop which might provide the model information about the camera orientation. This provides the model with information. By rotating the camera, it is made more difficult for the neural network to infer the sun's azimuthal angle.

Images are captured using the open-source Aravis viewer [15] (version 0.8.30). Utilizing the process of demosaicing, we proceed to extract the I_0, I_{45}, I_{90}, I_{135} intensity maps from the collected raw images.

Each instance in the dataset consist of the Stokes parameters I, Q, U. Equation 1 shows that the Stokes parameters can be retrieved using minimal pre-processing of the original intensity maps. Only +/- operations and rescaling by a factor 1/2 is necessary. For both the simulation and real data, the maps have been saved as 3 x 224 x 224 tensors. The 3 channels are for each Stokes' parameter respectively. This data shape is required to be used as input to ResNet-18 and many other standard convolution neural networks.

In order to investigate the effect of noise generated by clouds on the final error of the trained model, we used a 3x3 kernel on the AOP maps, which computes the standard deviation of the pixels contained in the kernel. This gives an estimate for the noisiness of an image, which corresponds well to how "cloudy" an image is, since clouds introduce noise into the AOP mapping [16]. Figure 8 shows this rudimentary approach can successfully classify cloudiness, although outliers still exist.

4 RESULTS

For ResNet, hyperparameter tuning is run for the real data with a train/test split of 85/15. The result of 5-fold cross-

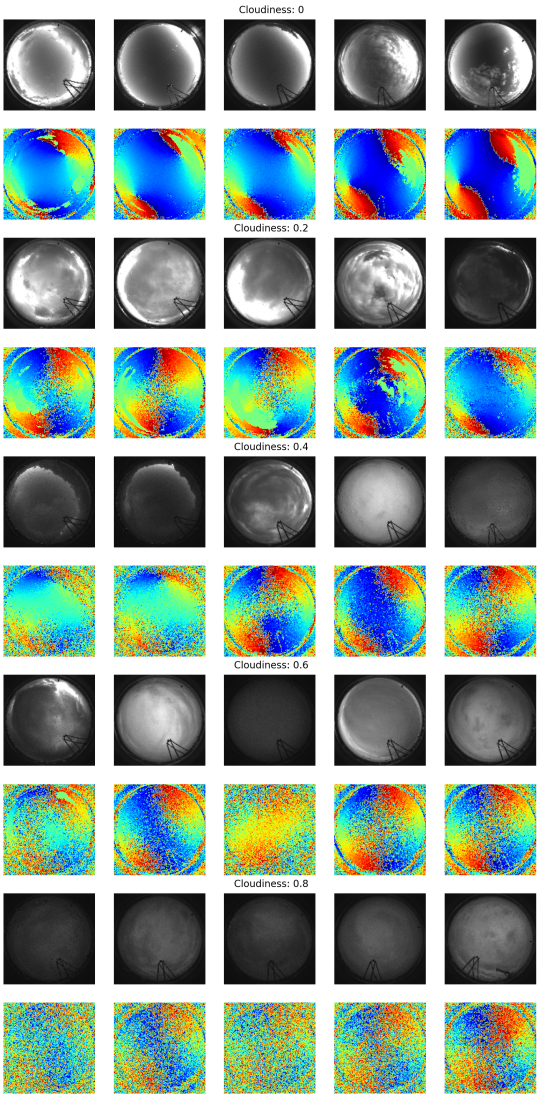


Figure 8: Example intensity maps, sorted by their cloudiness values

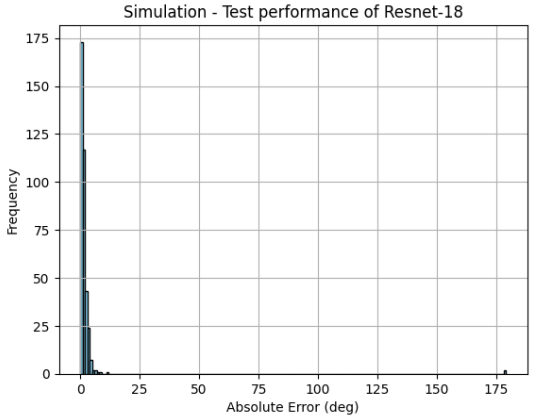


Figure 9: ResNet simulation error distribution

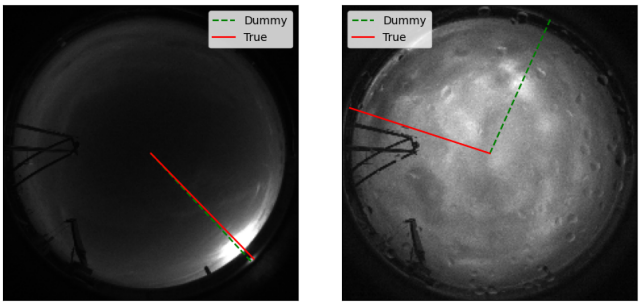


Figure 10: Brightest spot prediction on (left) a clear sky and (right) an overcast sky

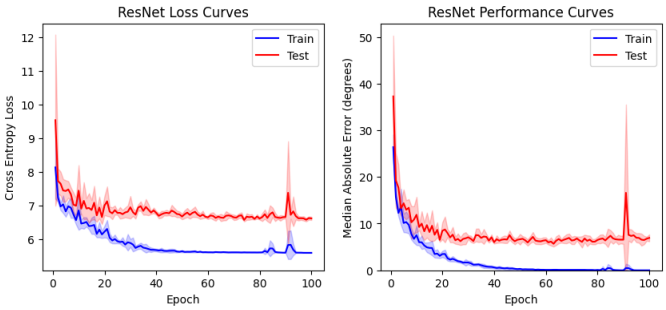


Figure 11: ResNet training curves on the real images

validation on the training set is shown in table 1. The selected parameters of lr = 5e-3 and m = 0.98 are used for the next subsections. The optimizer used is Adam with default PyTorch settings [17].

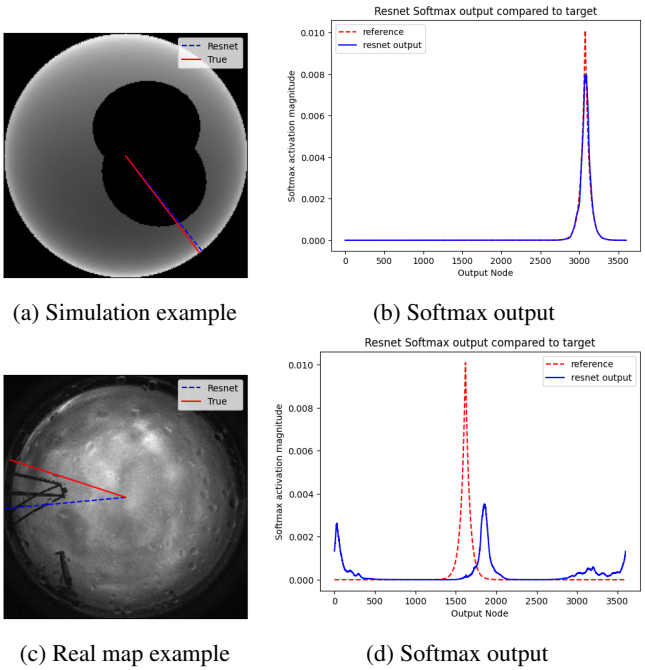
Learning rate	Exponential encoding decay, m			
	0.97	0.98	0.99	0.995
1e-4	29.86	22.72	26.29	26.83
5e-4	25.3	**21.81**	23.75	30.91
1e-3	30.69	24.60	27.68	24.02

Table 1: Average error (degrees) of different hyperparameters during gridsearch with a 65/20/15 train-val-test split after training for 50 epochs

Simulation

An augmented dataset of 2500 images is used as a basis for training the Resnet model using transfer learning [18]. The distribution of errors is visible in figure 9, with a final median absolute error of 1.00 degrees and a mean absolute error of 3.28 degrees.

To show the result of the model training, a simple visualization is shown in figure 12a. The Resnet model is able to mimic the exponential target output. This means the azimuth angle is accurately estimated, while being robust to the cloud mask.



(a) Simulation example (b) Softmax output

(c) Real map example (d) Softmax output

Figure 12: Examples of maps with their corresponding Softmax outputs. The top figures are simulated examples (a, b), and the bottom figures are real examples (c, d).

Real Data

Using the same train-test split of 85/15 that was used in the hyperparameter tuning, the model is trained on the dataset of 2618 real images. Figure 11 shows the learning curves, averaged over 5 training runs of 100 epochs each. The shaded area signifies a single standard deviation. The best performing model achieves a median absolute error of 4.30 degrees, and an mean absolute error of 12.10 degrees. The error distribution is visible in figure 15b. Figure 1 shows two example predictions of the trained ResNet model under different weather conditions.

We compare the robustness of the model by using a brightest spot predictor. This brightest spot model assumes the brightest spot in the image to be the sun as shown in figure 10. The best median error performance was found by pre-processing the images with a 33x33 Gaussian blur kernel. The resulting distribution of errors in shown in figure 15a. Figure 10 shows example predictions of the brightest spot model.

The brightest spot estimator performs worse than ResNet, with a median absolute error of 10.9 degrees and a mean absolute error of 27.24 degrees. The scatter plot in figure 14b and 14a shows the influence of cloud cover on both model performances. In general, a higher degree of cloudiness leads to both models estimating the solar meridian poorly (the brightest spot algorithm more so than the ResNet model). This reinforces the hypothesis of clouds introducing noise in polar-

http://www.imavs.org/

http://www.imavs.org/

ization information.

Looking at one such cloudy picture (figure 12c), the influence of the ResNet model becomes clear. The peak is lower when compared to 12a, which means the model is less certain about its prediction. Note that a second, lower peak is located 180 degrees from the predicted angle. Although the model correctly predicts the angle, this indicates the model is certain about the solar meridian orientation, but uncertain about the direction of the sun.

5 DISCUSSION

The accuracy of the model with simulated clouds is much higher compared to using real images. The median simulation accuracy is 1.0 degrees, whereas with real images it is 4.3 degrees. This highlights that the simulated clouds do not obscure polarization information in the same way that real clouds do. Another explanation for this difference in performance is the fact that the simulation produces maps in the global reference frame, where AOP symmetry is easily found.

From the plot in figure 14b, it is clear that from cloudiness levels above 0.7, the model accuracy is impacted. This demonstrates that our model is able to determine heading to some extent even with patchy skies, but it becomes increasingly difficult when there is less direct sky visibility.

In terms of performance, our model shows a clear advantage over the brightest spot algorithm, especially for cloudier skies. This indicates that our model is able to effectively estimate the solar meridian even in challenging conditions.

The collected dataset has limitations that need to be considered. The images have obstacles that obstruct the clear view of the sky, which can impact the accuracy of our model. These obstacles could have been used by the model as landmarks to help determine the solar angle. However, training the model with the objects all masked out with a circular mask, as shown in Figure 13, doesn't seem to significantly affect model performance. When trained on these masked images, even with a narrower field of view, the median error is 5.4 degrees compared to the original median error of 4.3 degrees. This shows that the model is in fact determining the heading from the polarisation information and not the obstacles. The Appendix shows the distribution of the azimuthal angles in our dataset is imbalanced. This might introduce bias in the model's performance.

6 CONCLUSION

In this paper, a polarization dataset has been created of skylight hemisphere pictures. The dataset contains 2618 images with various weather conditions. Additionally, a ResNet-18 model has been trained to perform solar azimuth estimation based on Stokes' parameters in the instrumental plane. The model attains a median absolute error of 1.0 degrees on simulation data with cloud masks, and 4.3 degrees on the real data. To compare, a brightest spot algorithm that predicts the direction of the brightest intensity achieves a me-

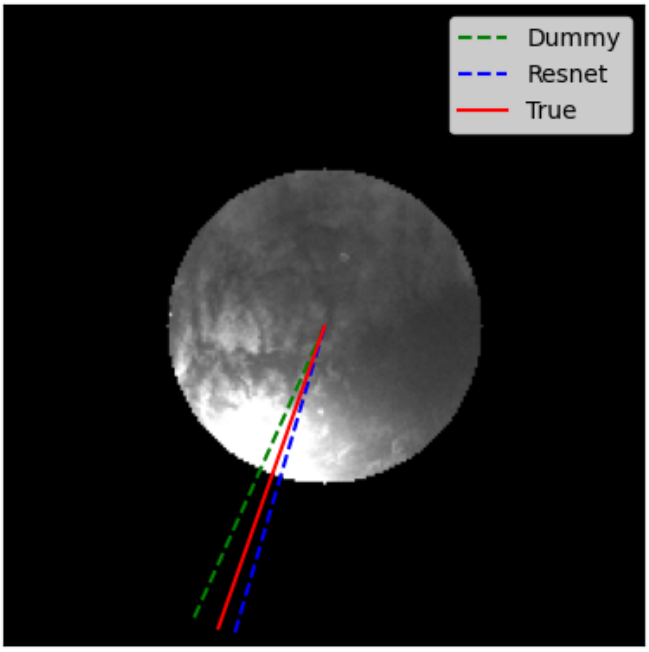
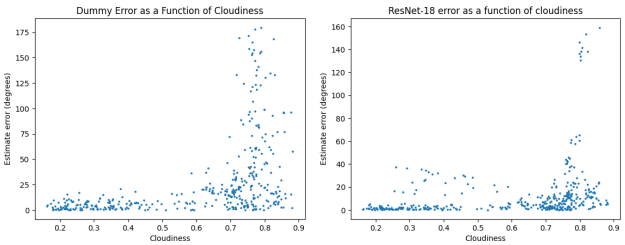
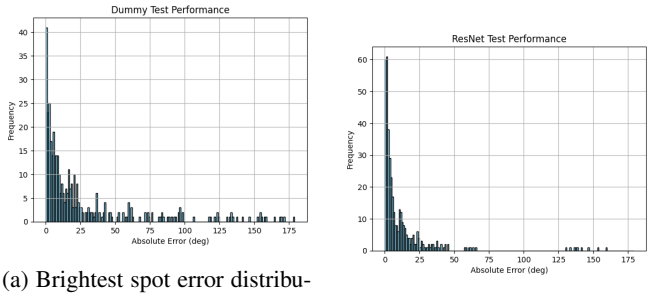


Figure 13: Masked image to determine effect of obstacles on model accuracy



(a) Brightest spot algorithm (b) ResNet

Figure 14: Influence of cloudiness on error



(a) Brightest spot error distribution (b) ResNet error distribution

Figure 15: Comparative error distributions

dian absolute error of 10.9 degrees on the real data. This research showcases that azimuthal angle estimation is possible with reasonable accuracy even under non-ideal weather conditions.

Future research is possible in expanding the polarization dataset to contain more images and improving the balance of the azimuth distribution. Also, a more refined metric for cloudiness could be developed to better quantify the influence of weather conditions on model performance. More advanced or custom machine learning models for azimuth estimation can be developed and validated on the dataset.

ACKNOWLEDGEMENTS

Setting up the physical setup would not have been possible without Stein Stroobants, Jesse Hagenaars, Erik van der Horst, Martin Søndergaard, and Bart Root.

DATA AVAILABILITY

The code and real dataset are both available upon reasonable request.

REFERENCES

[1] Kimon P Valavanis. Advances in unmanned aerial vehicles: state of the art and the road to autonomy. 2008.

[2] G Balamurugan, J Valarmathi, and V P S Naidu. Survey on uav navigation in gps denied environments. 2016 International conference on Signal Processing, Communication, Power and Embedded System (SCOPES), IEEE, 10 2016.

[3] D. Gebre-Egziabher and B. Taylor. Impact and mitigation of gps-unavailability on small uav navigation, guidance and control white paper. 10 2012.

[4] H. Liang, H. Bai, N. Liu, and X. Sui. Polarization navigation simulation system and skylight compass method design based upon moment of inertia. *Mathematical Problems in Engineering*, 204, 2020.

[5] H. Liang, H. Bai, K. Hu, and X. Lv. Polarized light sun position determination artificial neural network. *Applied Optics*, 61(6):1456, February 2022.

[6] G. Qian, J. Tang, C. Shen, and J. Liu. A highly robust polarization orientation method based on antisymmetry of skylight polarization pattern. *Measurement*, 2022.

[7] H. Zhao, W. Xu, Y. Zhang, X. Li, H. Zhang, J. Xuan, and B. Jia. Polarization patterns under different sky conditions and a navigation method based on the symmetry of the aop map of skylight. *Optics Express*, 26(22):28589–28603, 2018.

[8] L. M. Eshelman and J. A. Shaw. Visualization of all-sky polarization images referenced in the instrument, scattering, and solar principal planes. *Optical Engineering*, 2019.

[9] L. Guan, S. Liu, J. Chu, R. Zhang, Y. Chen, S. Li, L. Zhai, Y. Li, and H. Xie. A novel algorithm for estimating the relative rotation angle of solar azimuth through single-pixel rings from polar coordinate transformation for imaging polarization navigation sensors. *Optik*, 178:868–878, 2019.

[10] T. Kornland-Martinet, L. Poughon, M. Pasquinelli, D. Duché, J. R. Serres, and S. Viollet. Skypole—a method for locating the north celestial pole from skylight polarization patterns. *PNAS*, 120(30), 2023.

[11] X. Wu, C. Shen, D. Zhao, C. Wang, H. Cao, J. Tang, and J. Liu. Robust orientation method based on atmospheric polarization model for complex weather. *IEEE Internet of Things Journal*, 10(6):5268 – 5279, 2022.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[13] D. M. Stam, J. F. De Haan, J. W. Hovenier, and P. Stammes. Degree of linear polarization of light emerging from the cloudless atmosphere in the oxygen a band. *Journal of Geophysical Research*, 104(D14):16843 – 16858, 1999.

[14] H. Liang and H. Bai. Polarized skylight navigation simulation (psns) dataset. 2021.

[15] Aravis Project. Aravis.

[16] L. Shunzi, K. Fang, X. Han, G. Xiaohan, L. Haozhe, R. Yaohuang, C. Shouhu, and G. Yinjing. Biomimetic polarized light navigation sensor: A review. *Sensors*, 23(13), 2023.

[17] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. 2017.

[18] J.Plested and T.Gedeon. Deep transfer learning for image classification: a survey. May 2022.

APPENDIX A: DATASET DETAILS

The cloudiness values of the collected images are bimodal as the collection period was characterized by many mainly cloudy weeks followed by mainly clear skies as shown in figure 16.

Figure 17 shows that the azimuthal angles distribution in the dataset is very unbalanced. This will most likely introduce significant bias in trained models, so this is something to be aware of.

http://www.imavs.org/

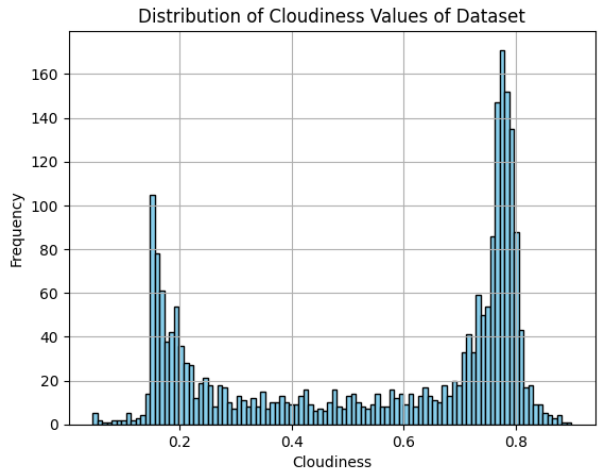


Figure 16: Distribution of cloudiness values in the dataset

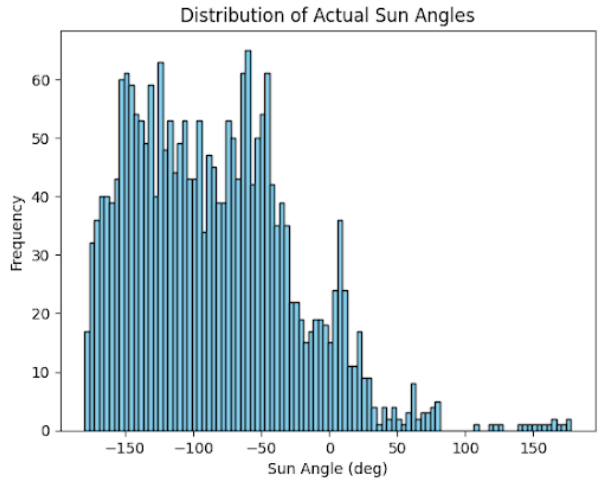


Figure 17: Distribution of azimuthal angles in dataset

http://www.imavs.org/

Pose Estimation with a Lightweight Visual-Inertial Neural Network for Agile UAV Flights

Kevin Guzman-Duran*[1],Alejandro Gutierrez-Giles[1] , and Jose Martinez-Carranza*[1]
[1]Instituto Nacional de Astrofisica, Optica y Electronica, Puebla, Mexico

http://www.imavs.org/

ABSTRACT

This paper presents a pose estimation system for UAVs designed for agile flights, utilizing the UZH-FPV Drone Racing dataset, which features high-speed, aggressive 6DoF trajectories for state estimation and drone racing. Our approach operates at 30 Hz, combining visual and inertial information through a sensor fusion technique incorporating temporal data into the input images. We use the lightweight convolutional neural network DeepPilot4Pose, which receives monocular images concatenated with inertial information. We process these data to calculate the UAV's position in real-time. Despite using a relatively small number of images for training compared to the trajectory length, our results show significant improvements in pose estimation accuracy and robustness in dynamic environments.

1 INTRODUCTION

In recent years, autonomous drones have grown exponentially in various fields, such as exploration, surveillance, supply delivery and cinematography. The ability of these Unmanned Aerial Vehicles (UAVs) to perform agile and precise manoeuvres in complex environments is crucial to their success in these applications. However, getting a UAV to navigate autonomously and efficiently in dynamic, high-speed scenarios presents a significant challenge due to the need for highly accurate and fast pose (position and orientation) estimation.

The central problem we are addressing is that UAVs, when flying at high speed and performing complex manoeuvres, demand pose estimation systems with high accuracy and low latency. However, traditional methods, such as GPS [1, 2] or additional hardware-based systems [3, 4, 5, 6, 7], are often not feasible due to their cost, weight, and power consumption, especially for small drones. Moreover, these methods could be more reliable in indoor or dense urban environments where GPS signals can be blocked or reflected [8, 2]. In such scenarios, there is an urgent need for a new, lightweight, efficient solution that can be easily integrated into UAVs and operate in real-time. Several methods on pose estimation for UAVs have been proposed in recent literature. For example, DeepPilot4Pose [9, 10], a compact convolutional neural network for visual pose estimation that runs onboard, flies in an indoor environment where no GPS or external sensors are available.

Our proposal presents a contribution to the field of pose estimation for UAVs. Our primary innovation lies in incorporating temporal and inertial data into the DeepPilot4Pose neural network, significantly improving pose estimation accuracy and robustness in dynamic environments. The DeepPilot4Pose network used in our approach has been optimized to fuse visual and inertial information at high speeds, providing real-time pose estimates at 30 Hz. This capability is crucial to enable the UAV to "see" and "feel" its environment through its cameras and sensors, using this information to calculate its position in real time.

Significantly, our system determines the absolute position of the UAV based on the vision system. The system can identify specific environmental features by leveraging visual information, ensuring precise localisation within a global reference frame.

We also evaluate our proposal using the UZH-FPV Drone Racing dataset [11], which provides images of aggressive flights and visual-inertial odometry data. Despite using a relatively small number of images for training compared to the trajectory length, our results show significant improvements in pose estimation accuracy and robustness in dynamic environments.

This paper is organised as follows: Section 2 provides a literature review on pose estimation; Section 3 describes the dataset used; Section 4 provides a description of the methodology for data collection, preprocessing and fusion; Section 5 describes the analysis of the results; and conclusions and future work are given in Section 6.

2 RELATED WORK

Visual-inertial odometry (VIO) and inertial measurement unit (IMU) based odometry have seen significant advances in recent years, driven by the need for accurate and robust pose estimation in various applications, such as robotics, autonomous vehicles, and augmented reality.

Visual odometry (VO) involves estimating the motion of a camera by analyzing captured image sequences. Early contributions in this field laid the foundation for modern VIO systems. [12] introduced a feature-based method for VO that uses feature matching and geometric triangulation to estimate

*Department of Computer Science at INAOE. Email addresses: {kevin.guzman, carranza}@inaoep.mx

camera motion. Their work established a framework for subsequent research in visual odometry.

Inertial odometry (IO) uses IMU data to estimate pose. Traditional methods, such as the Strap Inertial Navigation System (SINS), rely on double integration of acceleration data. However, these methods suffer from significant drift due to noise and sensor biases, making them unsuitable for long-term navigation tasks. The challenges posed by drift and noise in IMU data have motivated the development of more sophisticated algorithms that can provide accurate pose estimates over extended periods.

Geometric methods for VIO combine visual and inertial data using mathematical models and estimation filters. Among them is VINS-MONO by Qin [13], which employs a sliding window optimisation approach to fuse camera and IMU data. This method operates at approximately 10 Hz and is known for its robustness and accuracy, although it requires significant computational resources. OKVIS [14] integrates an extended Kalman filter (EKF) with sliding window optimisation, achieving high accuracy at a processing frequency of around 30 Hz at the expense of increased computational complexity. These methods have been instrumental in advancing the state of the art in VIO.

The advent of deep learning has led to the development of end-to-end odometry solutions that can learn complex representations directly from raw sensor data. VINet [15] is a pioneering work that frames VIO as a sequence-to-sequence learning problem using long short-term memory (LSTM) networks. This approach takes advantage of temporal dependencies in the data to improve the pose estimation accuracy, operating at a frequency of approximately 10 Hz. Similarly, IONet [16] employs LSTM networks to predict displacements from IMU data, focusing on 2D trajectory estimation with an operational frequency of about 10 Hz. AbolDeepIO [17] combines convolutional neural networks (CNNs) with LSTM networks to estimate 3D translational and rotational shifts from IMU data. This approach demonstrates significant improvements over traditional methods, with a processing frequency of approximately 15 Hz, showing the potential of deep learning in inertial odometry. These deep learning-based methods represent a shift toward leveraging large data sets and complex models to achieve superior performance under challenging conditions.

Hybrid methods combine geometric principles with deep learning techniques to leverage the strengths of both approaches. DeepAVO [18] introduces a four-branch network with spatial-channel attention to improve visual pose estimation accuracy, operating at approximately 20 Hz. This method integrates geometric constraints with deep learning models to improve robustness and accuracy. TLIO [19] represents another significant advance, combining a neural network-based displacement estimator with an EKF to achieve accurate inertial odometry. TLIO operates at around 50 Hz, highlighting the efficiency of tight integration between learned models and traditional filtering techniques.

Visual-inertial and inertial odometry landscape has evolved considerably, from early geometric methods to modern deep learning and hybrid approaches. Pioneering work laid the foundation for robust pose estimation, while recent developments have leveraged advanced learning techniques to push the limits of accuracy and robustness. Integrating deep learning with traditional geometric methods offers a promising direction for future research, aiming to achieve reliable and accurate odometry in a wide range of challenging environments.

By providing a comprehensive review of these advances, this section highlights key contributions and ongoing efforts in odometry, emphasizing the importance of combining geometric insights with data-driven models to improve the performance and applicability of odometry systems.

3 DATASET FOR EXPERIMENTAL STUDIES

This section delves into the dataset that forms the basis of our method's evaluation, shedding light on the algorithm's robustness. For agile UAV flight pose estimation, we leveraged the UZH-FPV Drone Racing dataset [11]. This dataset is distinguished by its high velocity and is purpose-built to capture daring and highly agile manoeuvres, making it a crucial resource for dynamic and demanding flight scenarios.

Inspired by the Autonomous Drone Racing challenge [20], the UZH-FPV Drone Racing Dataset, a widely used resource in visual-inertial odometry (VIO) applications, was developed by ETH Zurich and is publicly accessible. This dataset comprises grayscale images captured from a stereo camera mounted on a Micro Aircraft (MAV). It also includes simultaneous data from the accelerometer and gyroscope of the onboard IMU sensor, providing a comprehensive set of data for various analyses. Additionally, the dataset provides information on the calibration and noise values of the camera and IMU sensor, further enhancing its usability.

Using a high-speed data set such as the UZH-FPV Drone Racing is critical for developing and evaluating pose estimation algorithms capable of supporting agile UAV flight. The UZH-FPV Drone Racing Dataset recordings were conducted in indoor and outdoor environments, spanning different environments and camera orientations. The environments contain various objects to provide texture. The dataset is divided into easy, medium, and hard difficulty categories based on illumination, MAV speed, and image blur.

In our study, we used the following datasets: Indoor forward facing: 3-01, 6-02, 9-01, 10-01 and Indoor 45-degree downward facing: 2-01, 4-01, 9-01, 12-01, 13-02, 14-03. The first digit represents the trajectory, while 01, 02 and 03 indicate easy, medium, and hard difficulty categories.

To generate the UZH-FPV Drone Racing dataset, the researchers employed a first-person view (FPV) racing quadcopter equipped with sensors and piloted aggressively by an expert pilot. The MAV hardware included sensors for vi-

http://www.imavs.org/

sual and inertial measurements. IMU measurements were recorded at 500 Hz, while visual data were captured at 30 Hz. The IMU and cameras were synchronized so that half of the exposure coincided with the IMU measurements. A Leica Nova MS60 laser tracker, mounted on top of the MAV, was used to obtain reference data at a frequency of 500 Hz.

4 METHODOLOGY

In this study, visual and inertial information is evaluated in three different steps to perform pose estimation. These are the visual step, which uses the raw images in the data sets; the Inertial Image step, which combines the IMU data; and finally, the fusion step, which combines both information. Figure 1 shows the architecture of the proposed work.

4.1 Visual Data

Our study begins with the UZH-FPV Drone Racing dataset, a robust and reliable source of raw stereo images. We vertically concatenate the left mono and right mono images to form a single image. Then, we introduce our unique technique that incorporates the temporal dimension into the analysis through a three-dimensional concatenation of frames. This technique involves concatenating the current frames (n seconds) with frames captured at moments $n-5$ and $n-10$ seconds. The result is an RGB image where each channel represents a different time point, providing a rich representation of spatial and temporal information.

This concatenation allows us to capture relative motion between the camera and objects in the environment more accurately. The resulting images are used in the visual feature extraction steps, allowing the model to learn the dynamics of motion in the environment more effectively, as seen in Figure 1. In this way, the raw data from the camera is supplemented with motion information.

4.2 Inertial Data

The IMU sensors contain self-motion information, such as linear acceleration a and angular velocities ω. In some cases, the IMU also provides orientation information, e.g. roll-pitch-yaw angles (ϕ,θ,ϕ). These sensors, which are very sensitive to movement, operate at high frequency. In this study, the IMU frequencies for UZH-FPV Drone Racing are 500 Hz. In the data set, the IMU frequency is approximately 16 times the camera frequency. Therefore, there are 16 IMU data between any two frames.

In the inertial step, the goal is to use the raw IMU values between frames for pose estimation. In previous studies, time-dependent IMU raw data is generally directly fed to the LSTM layers, resulting in temporal IMU features. This study extracts features from inertial data differently from other studies to take advantage of the lightweight convolutional network architecture. Instead of using RNN-based LSTM as in traditional methods. We seek to use IMU data differently by relying on the robust estimation ability of CNN models.

$$\widehat{x} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

The accelerometer, gyroscope, and orientation values are first normalized to convert numerical IMU data into images due to their different lower and upper limits. Normalization adjusts these values to a standard range between 0 and 1. These normalized values are then multiplied by 255 to obtain a single-channel grayscale image with values ranging from 0 to 255. The resulting image comprises 144 pixels arranged in a 9×16 grid. Each pixel in this image represents the intensity of the corresponding accelerometer, gyroscope, or orientation value. In this way, the IMU data is transformed into a two-dimensional grayscale image, facilitating visualization and analysis and enabling better integration with the CNN models used in our study.

The same process as visual information is carried out to take advantage of temporality, i.e., incorporating the information corresponding to $n-5$ frames and $n-10$ into the remaining RGB channels.

4.3 Visual-Inertial Data

Finally, for data fusion we propose a process that involves the concatenation of visual information with inertial information. This three-dimensional concatenation of frames provides a representation that is rich in spatial and temporal information. The fusion of the frame and IMU information used in the fusion step and the temporal analysis representation are shown in Figure 2. Our preliminary results show a significant improvement in pose estimation when using both visual and inertial information, compared to using either one alone. The real power lies in the combination of both data sources, which allows us to capture the relative motion between the camera and objects in the environment more effectively, leading to accurate and robust pose estimation. The final image resulting from this fusion has a size of 224x224x3. These advanced data processing techniques ensure that our system can operate in real-time and quickly adapt to changes in the environment, significantly improving the accuracy and robustness of pose estimation in autonomous UAV navigation applications.

4.4 Pose Estimation

For 3D pose estimation, the DeepPilot4Pose [10] neural network is used. This network is capable of processing the fused images generated in the previous steps.

The DeepPilot4Pose network architecture comprises several convolutional layers that extract high-level features from the input images. These features are combined into fully connected layers to estimate the 3D position in the coordinates x, y and z. Additionally, a Savitzky-Golay filter [21] is applied to the output to smooth the pose estimates, thereby reducing noise and improving the estimation stability.

The DeepPilot4Pose network is a real-time powerhouse, processing the images at a frequency of 30 Hz, perfectly

http://www.imavs.org/

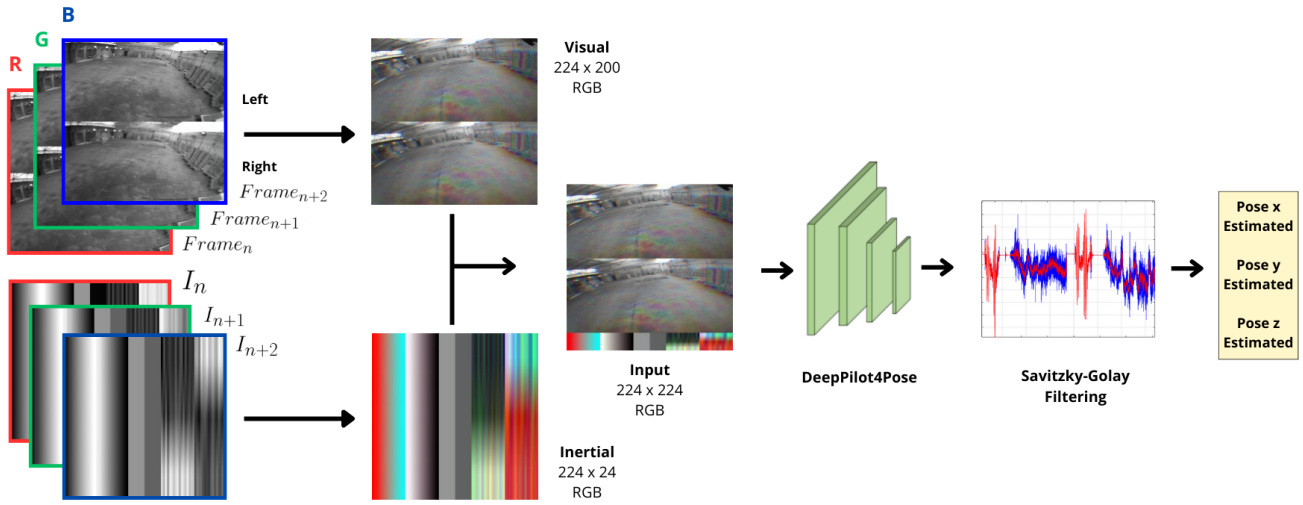


Figure 1: The architecture of the proposed approach.

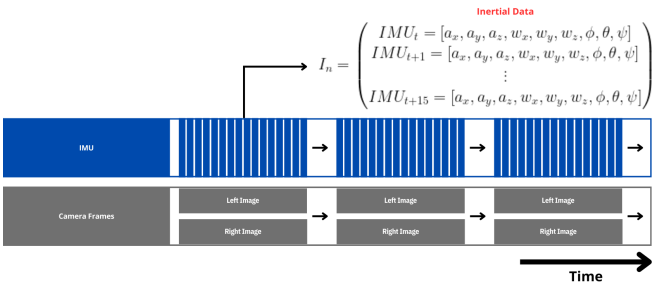


Figure 2: Temporal relationship between the IMU and the frames.

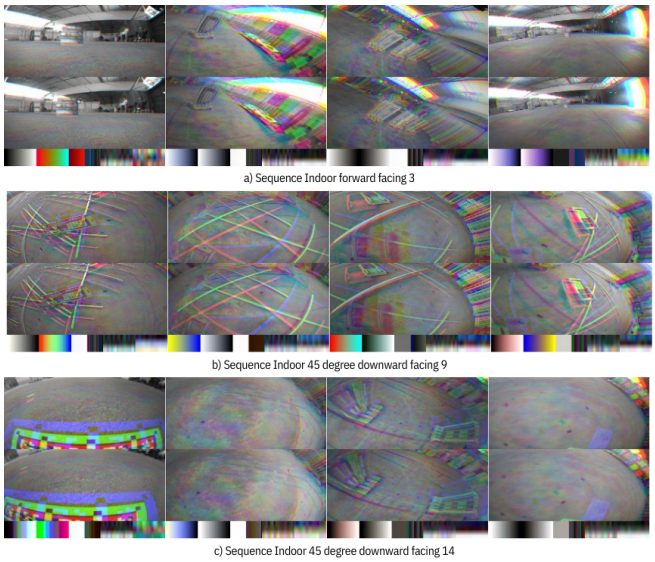


Figure 3: Illustration of a sequence of images representing the characteristics of a trajectory. The figure shows the complexity of the images, which can vary in texture from low to high, influencing the accuracy of pose estimation.

matching the camera image capture frequency. This capability ensures the system can provide accurate and timely pose estimates, making it an ideal tool for autonomous UAV navigation.

5 EXPERIMENTS AND RESULTS

This section analyzes the results of the proposed method applied to the UZH-FPV Drone Racing dataset. The training-test ratio for the data sets is estimated to be 80%-20%, taking at least one complete lap for training and another for testing. A desktop computer with NVIDIA GeForce RTX 3060 GPU was used during training and testing. The proposed algorithm calculates the average pose estimation time for the test data to be around 5 ms.

The root mean square error (RMSE) values are used at each of the three steps obtained from the test data Inside forward 03, 06, 09, 10 and Inside 45 degrees down 02, 04, 09,12, 13, 14. in this study, are given in Table 1.

In Table 1, the columns represent different aspects of the dataset and the results of the experiments. The columns include the maximum speed (V_{max}) of the UAV during the trajectory, the length of the trajectory in meters, the number of images used for training, the number of images used for testing, and the RMSE values for each method: Visual only, Inertial only, and the fusion of both.

It is evident that prediction performance generally decreases as the scene and flight conditions in the UZH-FPV Drone Racing dataset become more challenging. Factors such as image layout and UAV speed, which can negatively affect the texture information of the environment, are expected to hinder prediction performance under such conditions. The

http://www.imavs.org/

Sequences (Indoor)	$V_{\max}$ (m/s)	Length (m)	Training	Testing	Visual	Inertial	Fusion
Forward 3	9.50	22	1219	238	1.160	2.183	**0.864**
Forward 6	12.5	26	670	217	1.672	2.234	**1.623**
Forward 9	11.4	26	565	285	2.128	5.061	**0.967**
Forward 10	9.49	27	580	300	2.767	3.265	**2.136**
45 degree 2	6.97	27	1090	365	3.507	5.492	**3.387**
45 degree 4	6.55	24	840	200	1.750	5.514	**1.534**
45 degree 9	11.2	24	430	200	1.964	**1.535**	1.571
45 degree 12	4.33	28	590	597	**3.164**	5.384	3.294
45 degree 13	7.92	25	730	304	**1.214**	3.299	2.081
45 degree 14	9.54	22	815	170	1.414	**0.462**	0.867

Table 1: RMSE (m) values of the estimated UAV in the UZH-FPV Drone Racing dataset.

crux of the matter lies in using visual-inertial features together, ensuring minimal error due to fusion and providing more robust pose estimation in the face of these challenging conditions. The fusion results in Table 1 demonstrate that much more stable results are obtained than in the other two steps.

To better understand the performance of the UAV's predicted trajectory, the x, y, and z trajectories of the training data values and the position values for the test are visualised in Figure 4.

In this Figure 4, it is observed that when the test trajectory is similar to the one used in training and the greater the number of laps that represent the training, the performance in position estimation improves significantly. This occurs because the model has learned the specific trajectory and movement pattern characteristics during training, allowing it to make more accurate predictions in similar situations. However, due to the length and complexity of the trajectories, any slight difference can result in a decrease in performance since the model can only partially generalise to trajectories that deviate significantly from those observed during training.

Comparing the RMSE values for each experiment, it is observed that the best pose estimation is achieved using fused visual and inertial information. Visual information alone provides the second-best results, while inertial information alone is the least accurate. There are only a few cases where visual or inertial information obtains better results than information with the fusion of both sensors. This highlights the advantage of combining visual and inertial data for accurate posture estimation.

Additionally, it is essential to note that trajectories 45 degrees 9, 12, 13 and 14 show RMSE values where the highest values are not obtained by the fusion information, particularly in trajectories 45 degrees 9 and 14. These trajectories involve images with less texture and more repetition since the camera is tilted at 45 degrees and mainly captures the grey floor. This lack of distinguishing features makes it more challenging for the model to estimate the position accurately. As illustrated in the sequence images of Figure 3, these conditions can significantly affect the estimate's accuracy. In the case of sequence 14, the inertial information is reduced to the fusion threshold, highlighting that the maximum speed is 9.54 m/s, which leads us to think that due to the blurred information that captures the visual information, it hinders estimating the pose accurately.

Figures 5, 6, and 7 illustrate the obtained results, showing graphs that compare the trajectories of the actual position and the estimated position in the x, y, and z axes. Additionally, the results are compared for estimations using only inertial and visual information and the fusion of both. These comparisons allow us to evaluate the accuracy of each approach and highlight the significant improvement achieved by fusing visual and inertial data.

Finally, it is crucial to highlight that the DeepPilot4Pose neural network operates at 30 Hz. This high operating frequency is essential for real-time autonomous navigation applications, as it allows for rapid and accurate pose updates. Additionally, this frequency could be increased if images are captured at a higher rate, further improving the system's accuracy and responsiveness in dynamic environments.

6 CONCLUSION

In this work, we have addressed the problem of pose estimation in agile UAV flights, a critical task for applications in robotics and automation, especially in dynamic and challenging environments. Accurate position estimation is essential for autonomous navigation and executing complex manoeuvres in real time.

Our method, is based on the fusion of visual visual and inertial data through out the use of a compact Convolutional Neural Network. We carried out experimental tests using the well-known UZH-FPV Drone Racing dataset. The network employs a three-dimensional frame concatenation technique. By incorporating the temporal dimension into the analysis, this technique creates an RGB image where each channel represents a different time point. This unique representation, rich in spatial and temporal information, allows us to capture motion dynamics more effectively, thereby improving the accu-

http://www.imavs.org/

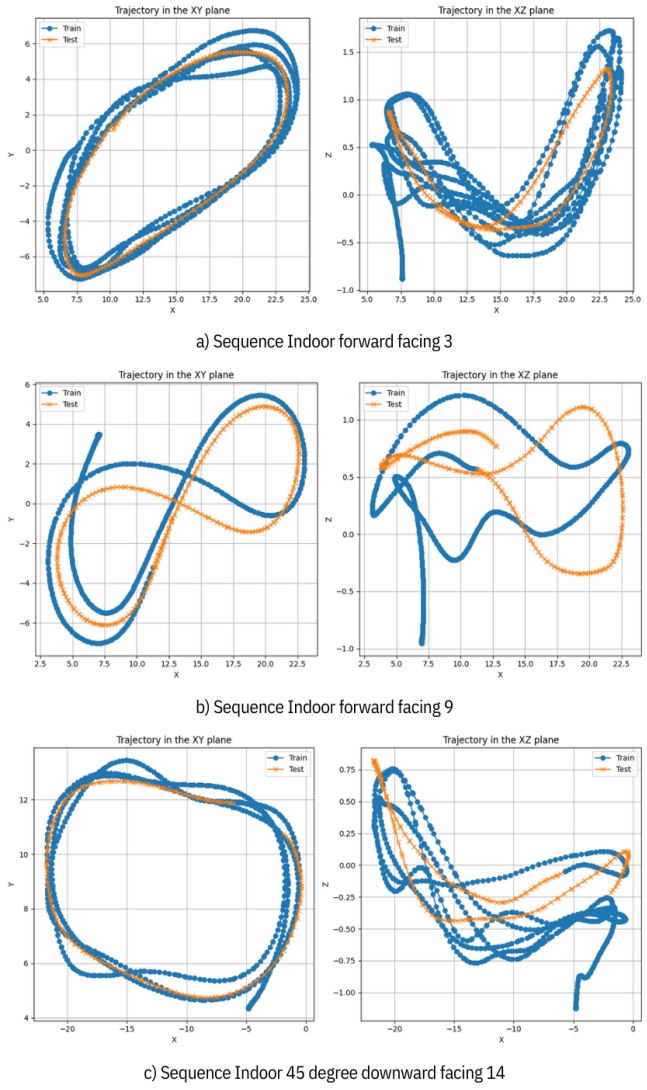


Figure 4: Comparison of two trajectories: a training trajectory and a test trajectory for pose estimation.

racy of pose estimation.

Our research has yielded tangible benefits, notably enhancing the robustness and accuracy of the pose estimation algorithm in the challenging scenarios captured in the UZH-FPV dataset. In terms of average error, our approach obtained an enhanced performance through out the fusion of the visual and intertial data in terms of accuracy. Another advantage is the lower memory consumption, while maintaining a processing time of 30 Hz, suitable for UAV applications with limited resources.

In future studies, we plan to evaluate our approach against other prominent methods mentioned in the literature review using the same UZH-FPV Drone Racing dataset. This will provide a broader perspective on our method's performance relative to other approaches, enhancing the robustness and reliability of our findings.

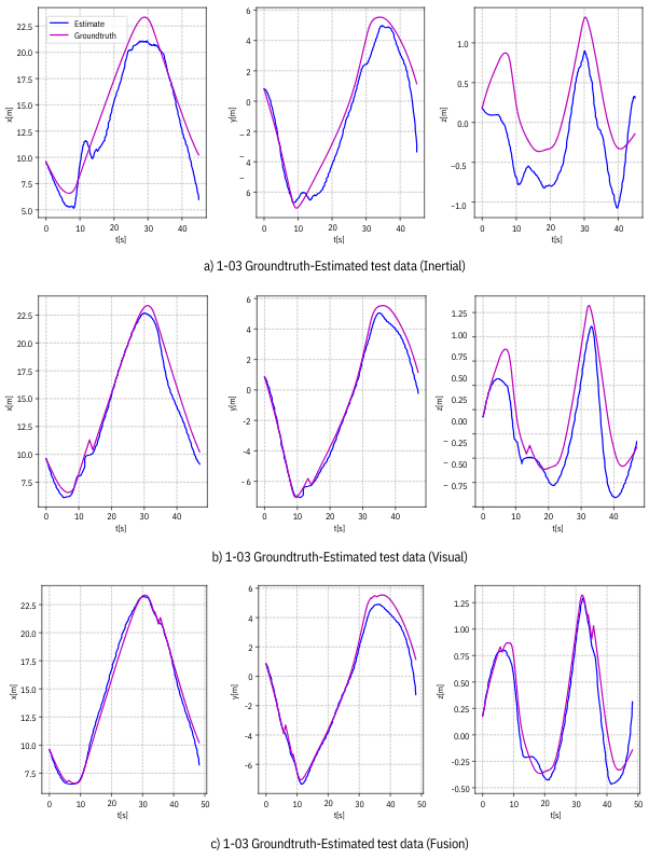


Figure 5: Groundtruth-estimated data from the Indoor forward facing 3 sequence.

Additionally, we consider several promising directions to expand and improve our research: to investigate techniques to improve the algorithm's robustness in adverse environmental conditions, such as variable lighting, changing weather conditions, and environments with dynamic obstacles.

REFERENCES

[1] Shah Zahid Khan, Mujahid Mohsin, and Waseem Iqbal. On gps spoofing of aerial platforms: a review of threats, challenges, methodologies, and future research directions. *PeerJ Computer Science*, 7:e507, 2021.

[2] Ganesan Balamurugan, J Valarmathi, and VPS Naidu. Survey on uav navigation in gps denied environments. In *2016 International conference on signal processing, communication, power and embedded system (SCOPES)*, pages 198–204. IEEE, 2016.

[3] John McConnell, Fanfei Chen, and Brendan Englot. Overhead image factors for underwater sonar-based slam. *IEEE Robotics and Automation Letters*, 7(2):4901–4908, 2022.

http://www.imavs.org/

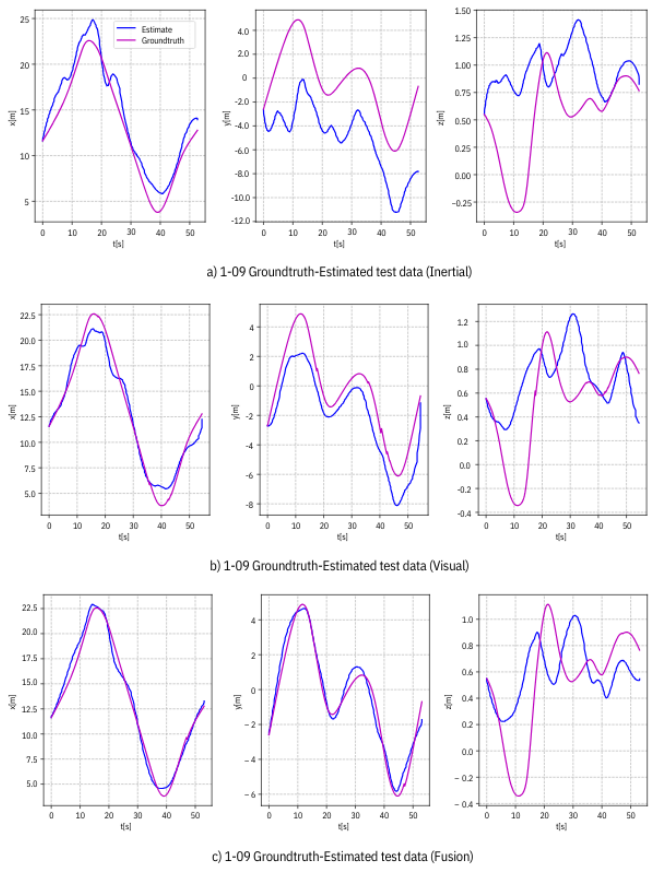


Figure 6: Groundtruth-estimated data from the Indoor forward facing 9 sequence.

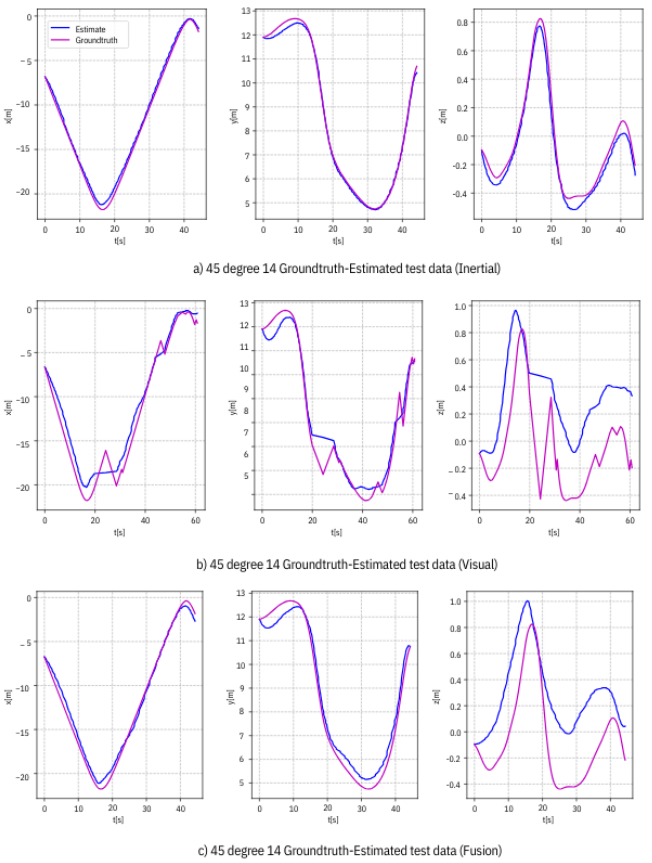


Figure 7: Groundtruth-estimated data from the Indoor 45 degree downward facing 14 sequence.

[4] Jie Qian, Kaiqi Chen, Qinying Chen, Yanhong Yang, Jianhua Zhang, and Shengyong Chen. Robust visual-lidar simultaneous localization and mapping system for uav. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5, 2021.

[5] Jung-Cheng Yang, Chun-Jung Lin, Bing-Yuan You, Yin-Long Yan, and Teng-Hu Cheng. Rtlio: Real-time lidar-inertial odometry and mapping for uavs. *Sensors*, 21(12):3955, 2021.

[6] Ahmed Elamin, Nader Abdelaziz, and Ahmed El-Rabbany. A gnss/ins/lidar integration scheme for uav-based navigation in gnss-challenging environments. *Sensors*, 22(24):9908, 2022.

[7] Matěj Petrlík, Tomáš Krajník, and Martin Saska. Lidar-based stabilization, navigation and localization for uavs operating in dark indoor environments. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 243–251. IEEE, 2021.

[8] Yueyan Zhi, Zhangjie Fu, Xingming Sun, and Jingnan Yu. Security and privacy issues of uav: A survey. *Mobile Networks and Applications*, 25(1):95–101, 2020.

[9] Leticia Oyuki Rojas-Perez and Jose Martinez-Carranza. Deeppilot: A cnn for autonomous drone racing. *Sensors*, 20(16):4524, 2020.

[10] Leticia Oyuki Rojas Pérez and Jose Martinez-Carranza. Deeppilot4pose: a fast pose localisation for mav indoor flight using the oak-d camera. *Journal of Real-Time Image Processing*, 20, 02 2023.

[11] Jeffrey Delmerico, Titus Cieslewski, Henri Rebecq, Matthias Faessler, and Davide Scaramuzza. Are we ready for autonomous drone racing? the UZH-FPV drone racing dataset. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2019.

[12] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. Ieee, 2004.

http://www.imavs.org/

[13] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[14] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015.

[15] Ronald Clark, Sen Wang, Hongkai Wen, Andrew Markham, and Niki Trigoni. Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[16] Changhao Chen, Xiaoxuan Lu, Andrew Markham, and Niki Trigoni. Ionet: Learning to cure the curse of drift in inertial odometry. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[17] Mahdi Abolfazli Esfahani, Han Wang, Keyu Wu, and Shenghai Yuan. Aboldeepio: A novel deep inertial odometry network for autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 21(5):1941–1950, 2019.

[18] Ran Zhu, Mingkun Yang, Wang Liu, Rujun Song, Bo Yan, and Zhuoling Xiao. Deepavo: Efficient pose refining with feature distilling for deep visual odometry. *Neurocomputing*, 467:22–35, 2022.

[19] Wenxin Liu, David Caruso, Eddy Ilg, Jing Dong, Anastasios I Mourikis, Kostas Daniilidis, Vijay Kumar, and Jakob Engel. Tlio: Tight learned inertial odometry. *IEEE Robotics and Automation Letters*, 5(4):5653–5660, 2020.

[20] Hyungpil Moon, Jose Martinez-Carranza, Titus Cieslewski, Matthias Faessler, Davide Falanga, Alessandro Simovic, Davide Scaramuzza, Shuo Li, Michael Ozo, Christophe De Wagter, et al. Challenges and implemented technologies used in autonomous drone racing. *Intelligent Service Robotics*, 12:137–148, 2019.

[21] Malek Karaim, Aboelmagd Noureldin, and Tashfeen B Karamat. Low-cost imu data denoising using savitzky-golay filters. In *2019 International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*, pages 1–5. IEEE, 2019.

http://www.imavs.org/

Evolving Spiking Neural Networks to Mimic PID Control for Autonomous Blimps

Tim Burgers
Delft University of Technology
Delft, The Netherlands

Stein Stroobants
Delft University of Technology
Delft, The Netherlands
s.stroobants@tudelft.nl

Guido C.H.E. de Croon
Delft University of Technology
Delft, The Netherlands
g.c.h.e.decroon@tudelft.nl

http://www.imavs.org/

***Abstract*—In recent years, Artificial Neural Networks (ANN) ave become a standard in robotic control. However, a signif- ant drawback of large-scale ANNs is their increased power onsumption. This becomes a critical concern when designing utonomous aerial vehicles, given the stringent constraints on ower and weight. Especially in the case of blimps, known or their extended endurance, power-efficient control methods re essential. Spiking neural networks (SNN) can provide a olution, facilitating energy-efficient and asynchronous event- riven processing. In this paper, we have evolved SNNs for ccurate altitude control of a non-neutrally buoyant indoor limp, relying solely on onboard sensing and processing power. he blimp's altitude tracking performance significantly improved ompared to prior research, showing reduced oscillations and a inimal steady-state error. The parameters of the SNNs were ptimized via an evolutionary algorithm, using a Proportional- erivative-Integral (PID) controller as the target signal. We eveloped two complementary SNN controllers while examining arious hidden layer structures. The first controller responds wiftly to control errors, mitigating overshooting and oscillations, hile the second minimizes steady-state errors due to non- eutral buoyancy-induced drift. Despite the blimp's drivetrain mitations, our SNN controllers ensured stable altitude control, mploying only 160 spiking neurons.**

***Index Terms*—Spiking Neural Networks (SNN), Leaky- ntegrate-and-Fire model, Neuromorphic Control**

I. INTRODUCTION

Throughout history, humans have been fascinated by how nimals gracefully and precisely navigate complex environ- ents. This fascination has inspired efforts to understand e brain's computational processes behind such behavior, eading to the development of Artificial Neural Networks ANN). These ANNs represent the neural processes using implified mathematical models. Their ability to approximate omplex non-linear functions makes them highly effective n controlling complex systems, such as quadrotors [1; 2]. lowever, as the size of ANNs grow, both response latency and omputational demands increase. The latter poses particular hallenges for robotic applications with restricted onboard nergy capacity, such as flying robots. A solution may be ound in the information transmission methods employed by iological brains. ANNs rely on continuous-valued signals, hereas the biological brain employs sparse spatial-temporal

This material is based upon work supported by the Air Force Office of cientific Research under award number FA8655-20-1-7044.

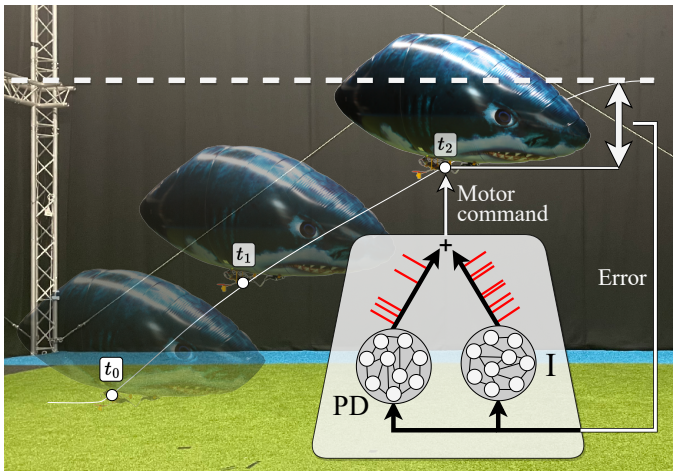


Fig. 1. The proposed SNN altitude controller for the blimp, where $[t_0,t_1,t_2]$ indicates time instances of the blimp's altitude while approaching a setpoint, marked by the dashed line.

"spike" signals—brief, rapid increases in neuron voltage—for data encoding and transmission.

There are neural networks that use this spike-based approach to transmitting information, called a Spiking Neural Network (SNN) [3]. These more biologically plausible neural networks show potential for energy-efficient and low-latency controllers [4]. This fact was demonstrated in a study by Vitale et al. [5], where an SNN controller in a fully neuromorphic control loop outperformed a conventional control loop on power consumption and control latency when tracking the roll angle of a bench-fixed 1 DoF birotor. However, the application of SNN controllers in robotics is still in its early stages of development. One of the biggest challenges is the availability of suitable training algorithms. The SNN's temporal dynamics, sparse spiking activity, and non-differentiable spike signal make most existing ANN training algorithms unsuitable [6]. Recent research has enabled the use of error backpropagation for SNNs by means of surrogate-gradient algorithms [7]. Nevertheless, training SNNs using these gradient-based algorithms is still difficult due to the susceptibility to local minima, exploding/vanishing gradient, and sensitivity to initial conditions [8].

Due to the increased training complexity of SNNs compared

http://www.imavs.org/

ɔ non-spiking counterparts, practical applications of SNNs ɪ the robotic control domain are still limited. Designing fully neuromorphic SNN controller to emulate low-level ontrollers, such as the Proportional-Integral-Derivative (PID) ontrol, still remains a complex task. Recent research showed NNs performing differentiation and integration within the etwork, by manually configuring neuron connections and /eights [9; 10; 11]. In these studies, the controllers were ɪplemented on Intel's Loihi neuromorphic processor, showasing the promise of neuromorphic hardware by exhibiting ery low latencies [12]. In Zaidel et al. [13], multiple popuations with pre-determined network parameters were used to ɪplement all three pathways of the PID controller to control 6 Degrees of Freedom (DoF) robotic arm. The integral athway was implemented using a fully recurrent population f neurons, while differentiation was achieved by using a low and fast time constant for two populations. Although ɪe integral controller succeeded in reducing the steady-state rror, it was unable to completely eliminate it. In another study, piking neurons were trained to replace the rate controller of tiny quadrotor, the Crazyflie [14]. Integration in the SNN ontroller was achieved through discretized Input-Weighted hreshold Adaptation (IWTA), where the threshold depends n the previous layer's spiking activity. The training process ad limitations because it relied partially on predetermined onnections and grouped network parameters.

In this work, we investigate the different mechanisms, recurɛncy and IWTA, used in prior SNN PID research that enabled ifferentiation and/or integration. For each mechanism, we volve an SNN to control the altitude of a real-world indoor limp. The blimp is an interesting test platform for the SNN ontroller, allowing validation of all components of the PID ontroller. The changing buoyancy over time requires a good ɪtegrator to be present. Moreover, due to the high delays and low system dynamics of a blimp, a high proportional gain s necessary in the reference PID controller. This reduces the limp's rise time, requiring a strong derivative in the conoller's output to prevent overshoot and oscillations. In Gonalez et al. [15], an open-source indoor blimp was designed nd used as a test vehicle for an evolved neuromorphic altitude ontroller, showing adequate tracking of the reference signal. lowever, even after including an additional non-spiking PD ontroller to the output of the SNN controller, there were still scillations present of approximately ± 0.3m. Additionally, ɪe SNN was only trained on a neutrally buoyant blimp. Slight hanges in the buoyancy of the blimp would cause a steady tate error, which the controller was unable to eliminate.

We build further on this research, presenting here the ollowing contributions: 1) We developed a fully neuromorphic eight controller for a blimp (visualized in Figure 1), using an volved SNN of only 160 neurons that is able to minimize the vershoot and oscillations while also removing the steady-state rror caused by the buoyancy of the blimp. 2) We analyze the ɪdividual and combined influence of recurrent connections nd IWTA on the performance of the SNN controller 3) We ɪade improvements to the hardware components of the open-source blimp, improving the onboard computational power and increasing the accuracy of the height measurements.

II. METHODOLOGY

The SNN controller consists of three layers of neurons, where all parameters are optimized using an evolutionary algorithm to mirror the output of a tuned PID controller. The Proportional-Derivative (PD) controller's rapid dynamics demand fast time constants, while the integral controller relies on slower dynamics and, thus, slow time constants. To facilitate the learning process, we split the controller into two separate parts based on the required time constants to model each component. After completing the training process, the evolved controllers are used to control the altitude of a helium-filled blimp. Detailed discussions on the SNN's structure, parameters, experiment setup, and the evolutionary training algorithm used in this study follow below.

A. Spiking Neural Network Controller

We use current-based Leaky-Integrate and Fire (LIF) neurons with a soft reset for the threshold (ϑ). The discretized equations that describe the dynamics of the three states of the neuron (synaptic current $i(t)$, membrane potential $v(t)$ and spike train $s(t)$) are described as follows:

$$i_i(t) = \tau_i^{\text{syn}} i_i(t-1) + \sum W_{ij} s_j(t) \quad (1)$$

$$v_i(t) = \tau_i^{\text{mem}} v_i(t-1) + i_i(t) - s_i(t-1)\vartheta_i \quad (2)$$

$$s_i(t) = H\left(v_i(t) - \vartheta_i\right) \quad (3)$$

where subscript i and j denote the post- and presynaptic neuron respectively. The discretized time constants, known as the decay parameters, of the synapses and the membrane potential are respectively referred to as τ^{syn} and τ^{mem}. The spiking behavior of a neuron is modeled using the Heaviside step function H, which outputs a spike when the membrane potential exceeds the threshold.

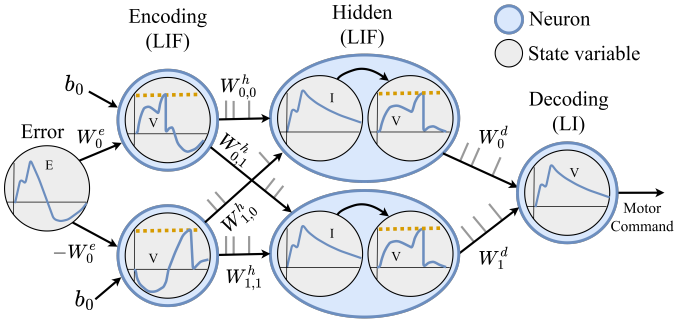


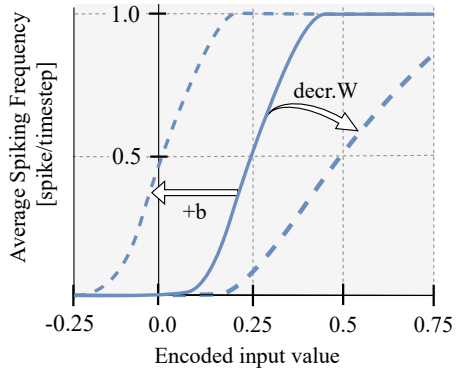
Fig. 2. The basic structure of the SNN controller. The encoding layer has an additional bias (b) added to the input current.

The basic structure of a spiking neural network controller is schematically depicted in Figure 2. The input weights, indicated by W^e, W^h and W^d, are linked to the encoding, hidden and decoding layers, respectively. The encoding layer is responsible for translating the floating-point input into

http://www.imavs.org/

sequence of spikes, and conversely, the decoding layer erforms the reverse operation.

The input to the network is the error of the controlled tate. After applying the encoding weights and biases, the rror signal is directly used as the synaptic current for the ncoding neuron (τ^{syn} = 0). To facilitate the training of the ncoding layer, we paired neurons with shared bias and flipped eight sign. This results in symmetric encoding, ensuring imilar spiking patterns for both positive and negative errors. he effect that the input weight and bias have on the spiking ehavior of a LIF neuron is presented in Figure 3. The ncoding layer incorporates a bias to achieve spike activity or the encoding of error values close to zero, which would therwise be impossible.



ig. 3. Influence of the input weight and bias on the spiking frequency of a IF neuron.

We study the effect of using different types of structures for ıe hidden layer of the SNN controller in this work. The most asic type of hidden layer (LIF) is depicted in Figure 2. In ddition to the basic structure, we evaluated the influence of ecurrency [13; 16] and Input-Weighted Threshold Adaptation [WTA) [14], as both these network structures demonstrated ıeir essential role in enabling integration within the SNN. We ocused solely on threshold adaptation linked to the incoming piking activity rather than the hidden neuron's activity itself. ın overview of the different hidden layer structures is shown ı Figure 4. In contrast to the original implementation in [14], ıe threshold for the IWTA-LIF neurons is modeled using a ecay term (τ^{th}), where the threshold converges back to a ase value (ϑ), after an increase/decrease (W^{th}) caused by an ıcoming spike. This method of implementing IWTA adds new ynamics to the threshold and increases the solution space.

The spiking neural network controller is decoded using a ingle leaky-integrator neuron, that calculates the exponential ıoving average of the spikes in the hidden layer.

. Real-World Experiment

To validate the performance of the SNN controller on real-world application, we implemented an SNN altitude ontroller for an open-source micro-blimp developed in [15]. he combination of the buoyancy-caused drift and slow ystem dynamics make the blimp a useful test vehicle for

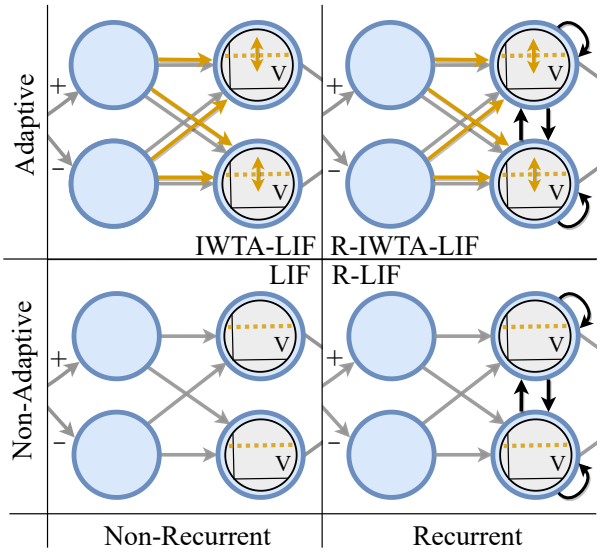


Fig. 4. Visualization of the different hidden layer structures. The two left and right circles represent the encoding and the hidden neurons respectively.

this research. The input of the SNN height controller is the difference between a reference altitude and the onboard lidar measurement, which makes it a fully on-board closed-loop control system. The output of the SNN is the motor command, $u \in [-3.3, 3.3]$, where $u < 0$ indicates downward and $u > 0$ upwards movement. The blimp's control system consists of two coreless direct current (DC) motors attached to a 180-degree rotating shaft, which enable the rotors to push the blimp both up- and downwards. A visual representation of the blimp and its hardware components is provided in Figure 5. Two improvements have been made to the blimp's hardware setup. Firstly, to improve the altitude tracking ability, we implemented a LiDAR sensor, the TFmini S LiDAR-module, which significantly increased the accuracy from $\pm$ 20cm to $\pm$ 1cm. Secondly, the Raspberry Pi Zero 2 W, running on Ubuntu 20.04, replaced its predecessor to increase the processing power and prevent software compatibility issues. The total weight of all hardware components attached to the blimp is 140g. In order to ensure smooth communication between all system components, we have used the Robot Operating System (ROS1) framework. The control loop runs at a rate of 10 Hz.

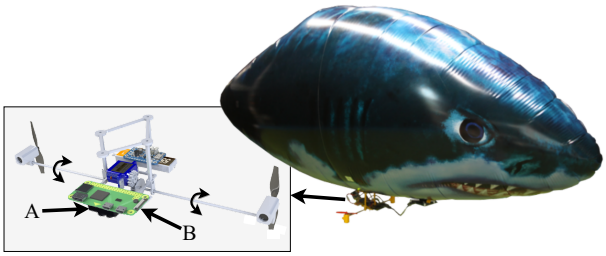


Fig. 5. The open-source micro-blimp used for the real-world experiments [15]. Two adjustments made on the blimp are: A) TFmini S LiDAR-module B) Raspberry Pi Zero 2 W.

http://www.imavs.org/

C. Evolutionary Training Algorithm

The training process for the SNN controller employs an evolutionary algorithm, consisting of two recurring steps: population creation and evaluation, with each cycle representing one generation of the evolution. In the first step, a set number of individuals forms the population, each representing a unique controller with slightly varying parameters. The second step ranks these individuals based on performance via a cost function, and this information guides the creation of the new population. Further details for each step are discussed below.

Every training loop was executed on the DelftBlue supercomputer, running on 20 cores for 50,000 generations with 50 individuals [17]. To prevent overfitting, we randomly sampled one of the 100 training datasets to evaluate each generation. Additionally, both the size and the frequency of the step inputs used in each dataset were varied to enhance the diversity of the training data. We conducted a total of 30 training loops for each combination of controller type (2) and hidden layer structure (4), resulting in a total of 240 training sessions. Each controller is limited to 80 neurons to showcase the potential of small-scale networks for neuromorphic control.

1) Population Creation: For the creation of a new population, we have used a Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [18]. CMA-ES is a distribution-based optimization algorithm that iteratively adjusts the mean and covariance matrix of a multi-variate Gaussian distribution, from which all network parameters of each individual are sampled.

The CMA-ES is implemented using the *Evotorch* Python library [19]. Every network parameter that needs to be evolved is initialized using the mean and standard deviation of a Gaussian distribution. Strict-bounded parameters, such as the decay constants in the neuron model, are constrained by rejecting and resampling. The initial mean is set by sampling from a uniform distribution within parameter bounds, while the standard deviation is set to 1/10th of the parameter's range. An overview of all trained parameters including the bounds is provided in Table I, where W^r represents the recurrent weights.

TABLE I
OVERVIEW OF ALL PARAMETERS AND BOUNDS USED TO EVOLVE THE SNN, WITH ADDITIONAL PARAMETERS IN THE HIDDEN LAYER FOR *RECURRENCY AND †IWTA

	Param.	Size	Bounds
Enc.	W^e	N	[-2, 2]
	b	N	[-1, 1]
	τ^{mem}	N	[0,1]
	ϑ	N	[0,10]
Dec.	W^d	N	[-1,1]
	τ^{mem}	N	[0,1]

	Param.	Size	Bound
Hidden	W^h	2N	[-2, 2]
	τ^{syn}	N	[0, 1]
	τ^{mem}	N	[0, 1]
	ϑ	N	[0, 10]
	$^*W^r$	NxN	[-1, 1]
	$^\dagger\tau^{th}$	N	[0,1]
	$^\dagger W^{th}$	NxN	[-1, 1]

2) Population Evaluation: The performance of each individual SNN controller is determined by comparing the output of the SNN controller (u) to the output of a tuned PD or integral controller ($\hat{u}$). The SNN is tasked to learn the mapping between the input signal (e) to the output of the target controller ($\hat{u}$). The discretized equation that describes the PID response is provided below:

$$\hat{u_k} = \underbrace{K_p e_k + K_d \frac{e_k - e_{k-1}}{T}}_{\text{PD controller}} + \underbrace{K_i (\sum_{k=0}^{k} T e_k)}_{\text{Integral controller}} \tag{4}$$

where K_p, K_i, and K_d refer to the proportional, integral and derivative gains respectively and T is denoted by the sampling period. Both the input error signal and the target controller response are recorded in a dataset.

To quantify the performance of the SNN compared to the target controller, the Mean Absolute Error (MAE) was used as the main term in the cost function. The MAE was used instead of the mean square error (MSE) to prevent over-penalization of the error that is created during the transient response to a step input because this led to more oscillations in the steady state. Additionally, the cost function was augmented with the Pearson Correlation Coefficient (PCC) [20], denoted by ρ, to incentivize that the sign of the output of the SNN and PD/I controllers is equal. The PCC measures the linear correlation between two signals, ranging from -1 to 1, where the latter indicates a fully linear relation. Since the fitness function is a minimization function, the PCC is included by adding $1 - \rho$ to the MAE. This results in the following cost function that was used in the population evaluation step of the evolutionary training process:

$$L(u, \hat{u}) = \text{MAE}(u, \hat{u}) + (1 - \rho(u, \hat{u})) \tag{5}$$

D. Dataset Generation

The methods used to gather the test and training data for each controller are discussed below.

1) PD controller: To successfully train the PD SNN controller, we need a broad spectrum of error signals. Therefore, we used a semi-randomly tuned PID controller on the neutrally buoyant real-world blimp. The error signal of these recordings is used as the SNN input data of the dataset. The target signal for the training algorithm is generated by passing the recorded error signal through a PD controller with the tuned gains for the blimp's altitude controller.

2) Integral controller: The Integral SNN has to learn to integrate the error within the SNN itself. For this training process, the decay parameter of the decoding neuron is purposely bound to ensure a quick decay (eg. [0-0.3]) in order to prevent the algorithm from converging to a slow decay. A slow decoding decay parameter would imply that the integration only happens in the decoding neuron, instead of in the hidden layer.

If the buoyancy remains constant throughout the training process of the integral controller, the network might learn to add a bias to counteract the buoyancy. Therefore we must adjust the buoyancy during training to ensure that the network learns to integrate information over time. Instead of recording multiple datasets with varying buoyancy, we decided to train the Integral controller on a model where we could also change

http://www.imavs.org/

ıe "buoyancy" within a single dataset to facilitate the learning rocess.

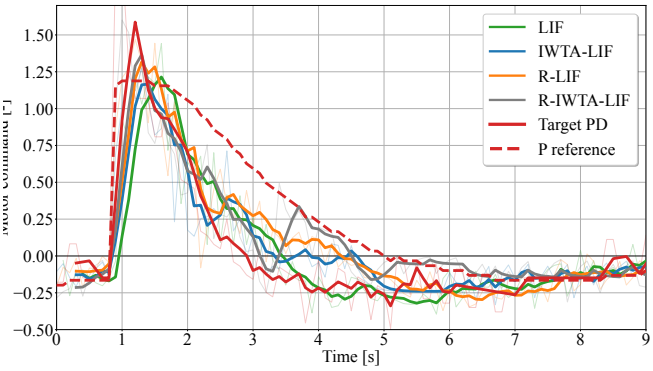
The blimp was modeled by the double integrator control roblem and controlled using a PID controller. The integral ain was set to match that of the tuned real-world blimp, while ıe PD gains were adjusted to align the model's dynamics pproximately with those of the tuned PID-Blimp system. In ıe double integrator system, the output of the controller, $u(t)$; directly proportional to the second derivative of the state lus an additional bias: $\ddot{x}(t) = u(t) + b$. The bias simulates ıe level of buoyancy of the blimp. Every time a step input ; received, the bias is randomly sampled ($U(-4, 4)$). Each ataset consists of 5 step inputs maintained for 50s.

III. RESULTS

The first subsection displays the training outcome of both NN controllers using a test dataset, followed by the assess-ıent of their performance on an actual blimp. The results also ontain a performance analysis of various neural mechanisms ithin the hidden layer.

. *Training of the SNN Controllers*

1) PD SNN Controller: The result of the PD training rocess is provided in Figure 6, showing a single step response om the test dataset. The solid red line represents the SNN's ırget, while the dashed red line depicts the proportional con-oller. The P controller is included to visualize the additional ffect of the derivative. Initially, all spiking PD controllers how clear influences of the derivative, as they match the target ignal. However, after the initial damping of the proportional utput, the controllers start to diverge. To prevent overshoot nd oscillations, the derivative controller should counteract ıe proportional controller when the state is approaching the etpoint, which happens around 4 seconds in the Figure. The nly controller that counteracts the P controller sufficiently ; the LIF SNN. Based on this analysis and the lowest loss alue across the entire test dataset, shown in Table II, we pted to use the LIF SNN for the blimp's altitude controller. 'he larger solution space for the recurrent and IWTA neuron tructures makes the search space more complex and leads to ocal minima.



ig. 6. The moving average of the step responses of all evolved SNN ontrollers compared to a PD target signal using a test dataset. The LIF shows erivative action by damping the command.

TABLE II
LOSS FOR SNN PD CONTROLLERS (u) ON COMPLETE TEST DATASET (500S) USING TUNED PD AS TARGET ($\hat{u}$)

	LIF	IWTA-LIF	R-LIF	R-IWTA-LIF
$L(u, \hat{u})$	0.68	0.74	0.74	0.73

2) Integral SNN Controller: The result of the integral training process is provided in Figure 7, showing the response of the different hidden layer mechanisms to a test dataset. The LIF SNN failed to learn to integrate, hence, it is excluded from the figure. All three spiking controllers following the test signal, without a clear standout performer. To see how well the controllers perform in the real-world, they are all tested on the indoor blimp.

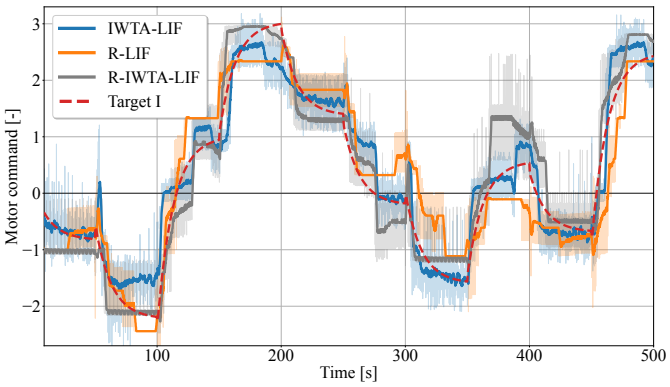


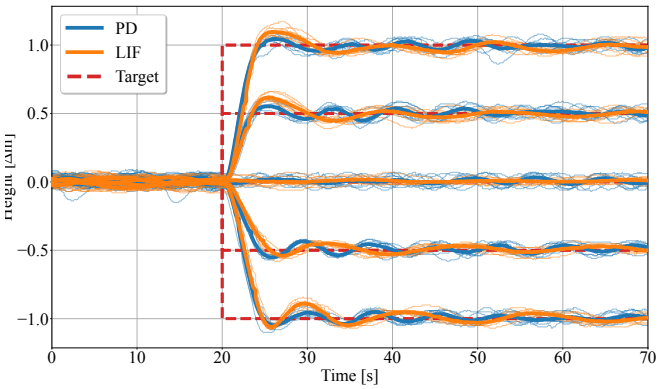
Fig. 7. The moving average of the step responses of three evolved SNN controllers compared to a integral target signal using a test dataset, with a changing bias every step input.

B. *Performance of SNN controlling Real-World Blimp*

To analyze the performance of each controller separately, we first test the PD SNN controller using a neutrally buoyant blimp. Afterward, we added some weight to the blimp to make it negatively buoyant. The negatively buoyant blimp is used first to evaluate the performance of the SNN I controllers, followed by the evaluation of the fully spiking controller.

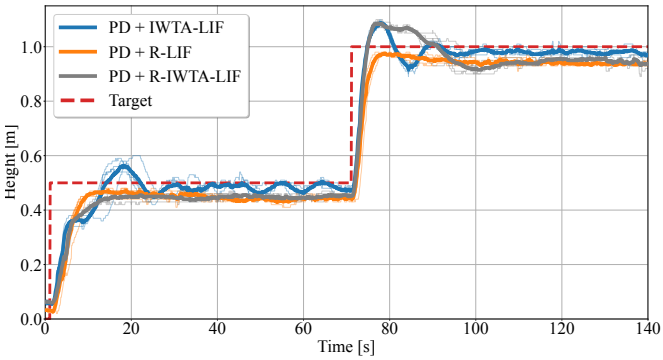
1) PD control of neutrally buoyant blimp: We assess the performance of the PD SNN controller with LIF hidden structure by comparing it to a tuned conventional PD controller. The tracking accuracy is tested using five different step sizes Δh=[1,0.5,0,-0.5,-1], maintained for 50s each and the results are shown in Figure 8. Both the conventional PD and the SNN show small oscillations, $\pm$6 cm, around the setpoint. These oscillations are caused by the discretized mapping of the motor command to the actual voltage sent to the motors using PWM. This causes a deadzone to be present in the motor command signal, which is the region of the motor commands, u =[-0.1,0.1], that does not result in the actuation of the rotors.

2) Integral control of negatively buoyant blimp: To isolate the effect of the spiking integrator, we used the non-spiking PD controller in combination with the spiking integrator to control the negatively buoyant blimp. The result of two-step responses



ig. 8. Real-world step responses of the altitude of the neutrally buoyant limp using a conventional PD control and an SNN controller with no ecurrency or IWTA in the hidden layer. Each setpoint was tested eight times or every controller and the average is shown by the thick line.

s shown in Figure 9, where the setpoint was changed after 70 econds. When analyzing the steady-state error, we take the verage over the last 10 seconds of each step. The steady-state rror of the IWTA-LIF (±2 cm) is significantly smaller than ne ones for the R-LIF and R-IWTA-LIF (±5 cm). Despite ne small oscillations caused by the LIF SNN, we decided to se this spiking integrator for the full spiking controller due o the minimal steady-state error.



ig. 9. Real-world step responses of the altitude of the negatively buoyant limp using a conventional PD controller and different spiking integrators. he average over five runs is shown for each controller.

3) Full spiking control of negatively buoyant blimp: The ombination of both the spiking PD (LIF) and integral (IWTA-IF) controller is shown in Figure 10. We analyzed the step response for the blimp using different setpoints h=[0.5,1,1.5]m, naintained for 70s. The combined SNN controller demonstrates effective altitude control while minimizing the steady-tate error to ±3cm. However, the SNN controller shows elatively large initial oscillations when receiving a downward tep input, compared to the upward steps. The oscillations esult from the delay introduced when the rotors must make a 80-degree turn during direction changes. Given the blimp's egative buoyancy, continuous upward thrust is essential for tability. In cases of upward step inputs, the rotors constantly push the blimp upwards. Conversely, when a lower setpoint is used, the blimp is initially pushed downwards by the rotor before pushing back up to attenuate the movement. This difference in overshoot is also visible for the PID controller, with an average overshoot of 14cm for upward steps and 20cm for downward steps.

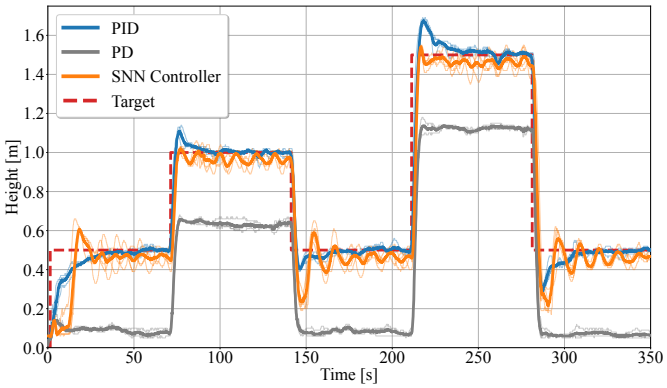


Fig. 10. Real-world multi-step response of the altitude of the negatively buoyant blimp using a non-spiking PID & PD controller and the SNN controller, which is the combination of the SNN PD (LIF) and the SNN integral (IWTA-LIF) controller. The average over five runs is shown for each controller.

IV. CONCLUSION

In this work, we evolved a spiking neural network (SNN) that successfully controls the altitude of a non-neutrally buoyant indoor blimp. The SNN parameters were optimized through an evolutionary algorithm, facilitating extensive exploration of the solution space. This exploratory training approach allowed for an in-depth analysis of various hidden-layer configurations, recurrency and Input Weighted Threshold Adaptation (IWTA), for the Leaky-Integrate and Fire (LIF) neuron model. As a result, we developed two complementary SNN controllers which, when combined, achieved accurate tracking of the reference state. The first controller exhibited rapid response to control errors while effectively mitigating overshoot and large oscillations, after being trained on a tuned PD controller for the blimp. In parallel, the second controller was designed to minimize steady-state errors arising from the blimp's non-neutral buoyancy-induced drift. This controller learned to perform integration of the error using IWTA within the hidden layer of the network.

Despite the limitation within the blimp's current drivetrain configuration, the developed SNN controllers have showcased their ability to maintain stable altitude control, employing just 160 spiking neurons. All processing and sensing is performed onboard the blimp, with the SNN running on the Raspberry Pi's CPU. Future research will focus on the completion of the neuromorphic control loop, integrating event-based sensors with neuromorphic processors. This integration aims to fully demonstrate the potential of neuromorphic computing in robotic control.

http://www.imavs.org/

http://www.imavs.org/

REFERENCES

[1] D. Zhang, A. Loquercio, X. Wu, A. Kumar, J. Malik, and M. W. Mueller, "A Zero-Shot Adaptive Quadcopter Controller," 9 2022.

[2] M. Heryanto, H. Suprijono, B. Yudho, and B. Kusumoputro, "Attitude and altitude control of a quadcopter using neural network based direct inverse control scheme," *Advanced Science Letters*, vol. 23, pp. 4060–4064, 05 2017.

[3] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[4] Z. Bing, C. Meschede, F. Röhrbein, K. Huang, and A. C. Knoll, "A survey of robotics control based on learning-inspired spiking neural networks," *Frontiers in Neurorobotics*, vol. 12, 2019.

[5] A. Vitale, A. Renner, C. Nauer, D. Scaramuzza, and Y. Sandamirskaya, "Event-driven Vision and Control for UAVs on a Neuromorphic Chip," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2021-May, pp. 103–109, Institute of Electrical and Electronics Engineers Inc., 2021.

[6] A. Taherkhani, A. Belatreche, Y. Li, G. Cosma, L. P. Maguire, and T. M. McGinnity, "A review of learning in biologically plausible spiking neural networks," *Neural Networks*, vol. 122, pp. 253–272, 2020.

[7] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, pp. 51–63, 11 2019.

[8] Z. Bing, C. Meschede, F. Röhrbein, K. Huang, and A. C. Knoll, "A survey of robotics control based on learning-inspired spiking neural networks," *Frontiers in Neurorobotics*, vol. 12, 2019.

[9] R. K. Stagsted, A. Vitale, A. Renner, and L. B. Larsen, "Event-based PID controller fully realized in neuromorphic hardware: A one DoF study," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2, pp. 10939–10944, 7 2020.

[10] R. K. Stagsted, A. Vitale, A. Renner, and L. B. Larsen, "Towards neuromorphic control: A spiking neural network based PID controller for UAV," in *Robotics: Science and Systems XVI*, vol. 1, Robotics: Science and Systems Foundation, 1 2020.

[11] S. Stroobants, J. Dupeyroux, and G. De Croon, "Design and implementation of a parsimonious neuromorphic PID for onboard altitude control for MAVs using neuromorphic processors," in *ICONS'22 Proceedings*, vol. 2657, pp. 1–9, CEUR-WS, 2022.

[12] M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. K. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. H. Weng, A. Wild, Y. Yang, and H. Wang, "Loihi: A Neuromorphic Manycore Processor with On-Chip Learning," *IEEE Micro*, vol. 38, pp. 82–99, 1 2018.

[13] Y. Zaidel, A. Shalumov, A. Volinski, L. Supic, and E. Ezra Tsur, "Neuromorphic NEF-Based Inverse Kinematics and PID Control," *Frontiers in Neurorobotics*, vol. 15, 2 2021.

[14] S. Stroobants, C. De Wagter, and G. C. H. E. de Croon, "Neuromorphic Control using Input-Weighted Threshold Adaptation," vol. 1, Association for Computing Machinery, 2023.

[15] M. Gonzalez-Alvarez, J. Dupeyroux, F. Corradi, and G. C. De Croon, "Evolved neuromorphic radar-based altitude controller for an autonomous open-source blimp," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 85–90, 2022.

[16] H. Qiu, M. Garratt, D. Howard, and S. Anavatti, "Towards crossing the reality gap with evolved plastic neurocontrollers," in *GECCO 2020 - Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, vol. 1, pp. 130–138, Association for Computing Machinery, 6 2020.

[17] Delft High Performance Computing Centre (DHPC), *DelftBlue Supercomputer (Phase 1)*, 2022. https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1.

[18] N. Hansen, "The CMA Evolution Strategy: A Tutorial," pp. 1–39, 2016.

[19] N. E. Toklu, T. Atkinson, V. Micka, P. Liskowski, and R. K. Srivastava, "EvoTorch: Scalable Evolutionary Computation in Python," pp. 1–25, 2023.

[20] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson Correlation Coefficient," in *Noise Reduction in Speech Processing*, pp. 1–4, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

Accelerating Swarms : Harnessing Hardware Acceleration and Parallelization in Multi-Agent Reinforcement Learning

T. Gavin[*x+], S. Lacroix[x], and M. Bronz[+]
[*]*IAS*, *Thales LAS*, Rungis, France
[x]*RIS*, *LAAS-CNRS*, Toulouse, France
[+]*Fédération ENAC ISAE-SUPAERO ONERA*, *Université de Toulouse*, France
timothee.gavin@thalesgroup.fr , simon.lacroix@laas.fr , murat.bronz@enac.fr

http://www.imavs.org/

ABSTRACT

In Multi-Agent Reinforcement Learning, researchers often face a challenging trade-off: the use of complex environments that demand substantial computational resources, or simpler dynamics for expedited execution, albeit at the cost of transferability to more realistic tasks. This article delves into the potential of vectorizable environments, which enable parallel environment rollout and fully harness the parallelization capabilities of modern GPUs. We present a comparison of various RL environment libraries, highlighting their features and limitations for end-to-end hardware-accelerated training pipelines. We observe that the most commonly used RL algorithm libraries have yet to fully embrace end-to-end hardware-accelerated training pipeline and the limited cross-compatibility between the frameworks used for hardware acceleration and parallelization in machine learning: PyTorch, TensorFlow, and JAX, limits the mix-and-match options for RL environments and algorithms libraries.

1 INTRODUCTION

A swarm of drones is a prime example of the Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs) problem that can be addressed using Multi-Agent Reinforcement Learning (MARL). In this application, each drone operates as an individual agent with its own set of actions and observations. The drones must collaborate to achieve a common goal, such as area surveillance or delivery tasks, despite having only partial and local observations of the environment. The inherent non-stationarity and decentralized partial observability in such a setup makes MARL a suitable approach. Through MARL, each drone learns from its interactions with the environment and other drones, improving its policy over time to contribute effectively to the overall objective of the swarm. However, generating environment data can be slow, especially for complex environments.

Reinforcement Learning (RL) is notoriously sample inefficient, a problem worsened in MARL due to non-stationarity and decentralized partial observability. Workloads for RL agents are steadily increasing as we tackle more complex problems that require millions of training steps. This issue is worsened by the need for hyperparameter tuning in deep RL, which requires time-consuming repeated training sessions.

To address this issue, research has focused on scalable RL frameworks. Current training pipelines use CPUs for simulating environment physics, calculating rewards, and running the environment, while GPUs accelerate neural network models during training and inference. Increasing environment throughput often involves parallelizing the environment; prior work has used multiple CPU cores and multi-threading to run multiple environment instances in parallel to speed up rollouts. However, this approach is limited by the number of cores and memory usage. Therefore, training in complex environments requires scaling up on massive distributed systems, such as large clusters with thousands of CPUs. Despite the use of these powerful systems, training times remain lengthy due to the data transfer bottleneck between the CPU, where the environment is simulated, and the GPU, where the agents are trained and evaluated. This has led to a trade-off in RL research: use complex environments requiring large compute resources, making it inaccessible to researchers with limited resources, or rely on simpler dynamics for faster execution, which makes the transferability of results to more realistic tasks more challenging.

To overcome these challenges, researchers are exploring the use of hardware accelerators to create end-to-end training pipelines. This approach leverages the parallelization capabilities of modern GPUs to achieve orders of magnitude faster training pipelines. This could significantly advance RL research by speeding up the testing and iteration of ideas, lowering computational barriers for in-depth MARL research, and allowing experiments that previously required a data center to be conducted locally on desktops or small GPU servers.

However, this article aims to show that the most commonly used RL libraries have yet to fully embrace end-to-end hardware-accelerated training pipelines. Hardware acceleration in the field of machine learning (ML) necessitates

http://www.imavs.org/

a choice between the three leading frameworks: PyTorch [1], TensorFlow [2], and JAX [3], which have limited cross-compatibility. This means that if an RL environment library adopts one of these frameworks, it may face compatibility issues with RL algorithms that are designed using the other frameworks. This limits the mix-and-match options for RL environments and algorithms libraries.

2 BACKGROUND

2.1 Reinforcement Learning

Reinforcement Learning is a type of machine learning for sequential decision-making. In a *rollout* phase, an *agent* interacts with an uncertain *environment* which provides it with a *partial observations* of its state, takes a series of *actions* following a *policy* and receives a scalar feedback in the form of *rewards*. These sequences of observe-act-reward, repeated over time, form the *rollouts*. The collected rollouts are then used to update the policy in a learning phase, which will then be employed in the rollout phase of the next training iteration.

2.2 Multi-Agent Reinforcement Learning

Single-agent RL methods often fail in multi-agent settings due to the well-known curse of dimensionality and non-stationarity. Recently, MARL approaches addressed these issues with the Centralized Training, Decentralized Execution (CTDE) approach. By training the agents in a centralized manner to leverage global information and then executing the learned policies in a decentralized manner, CTDE ensures scalability and robustness. Several state-of-the-art MARL algorithms have been developed under the CTDE framework, including value-based methods like QMIX [17] and VDAC [18], and policy gradient methods like MADDPG [19], MAPPO [20], and multi-agent versions of SAC [21].

2.3 Environments

Simulations are integral to RL for ensuring safety and enhancing iteration speed. They provide a risk-free testing ground, preventing real-world mishaps like crashes or environmental damage. Additionally, simulations are efficient and scalable, enabling rapid trial-and-error iterations, which are essential for the learning process in RL. A wide range of simulators were designed to help develop RL and MARL algorithms and solve problems for real-life drone applications. The degree of realism of these environments can vary. On one end, we have environments with complex physics engines that simulate low-level tasks, focusing on issues related to sensing and low-level control. On the other end, we have high-level simulations with often 2D dynamics engines that focus on high-level tasks like coordination for multi-agent operations. Table 1 lists a set of environments and highlights their features related to drone-oriented tasks.

3 THREE ML FRAMEWORKS

Hardware acceleration in the field of machine learning, whether for environments or training algorithms, necessitates a choice between the three leading frameworks: PyTorch [1], TensorFlow [2], and JAX [3]. In this section, we describe each framework. Table 2 summarizes the various features of each framework.

Environment	State[a]	Action[b]	PartObs[c]	Random[d]	Physics[e]	Drone[f]	MARL[g]	#Agents[h]	Collab[i]	Advers[j]	Hetero[k]	Comm[l]
gymnasium [4]	C	C	✓	✓	Any	✓	✗	-	-	-	-	-
PettingZoo [5]	C	C	✓	✓	2D	✗	✓	<100	✓	✓	✓	✓
Pybullet-drones [6]	C	C	✗	✓	3D	✓	✓	<10	✓	✗	✗	✗
EnvPool [7]	C	C	✗	✓	2D/3D	✗	✗	-	-	-	-	-
MAgent [8]	D	D	✗	✓	Grid	✗	✓	>1000	✗	✓	✗	✗
brax [9]	C	C	✗	✓	3D	✗	✗	-	-	-	-	-
gymnax [10]	C	C	✗	✓	2D	✗	✗	-	-	-	-	-
IsaacSIM [11]	C	C	✓	✓	3D	✓	✓	<10	✓	✗	✗	✗
Gigastep [12]	C	C	✓	✓	2D	✗	✓	>1000	✓	✓	✓	
VMAS [13]	C	C	✓	✓	2D	✗	✓	<100	✓	✓	✓	✓
JaxMARL [14]	C	C	✓	✓	2D/3D	✗	✓	<10	✓	✓	✓	✓
Jumanji [15]	C	C	✗	✓	2D	✗	✓	-	✓	✓	✗	✓
CrazyRL [16]	C	C	✗	✗	3D	✓	✓	<10	✓	✗	✗	✗

[a] Continuous (C) or discrete (D) state space
[b] Continuous (C) or discrete (D) actions space
[c] Environments support partial observability
[d] Environments support domain randomization
[e] Type of physics engine used for the simulation
[f] Environment natively handle drone oriented tasks
[g] Native support for multi-agent reinforcement learning
[h] Number of agents generally supported
[i] Collaborative multi-agent tasks
[j] Adversarial multi-agent tasks
[k] Handle heterogeneous agents
[l] Handle Inter-Agents communications

Table 1: Comparison of different RL environments features

Framework	PyTorch	TensorFlow	JAX
Neural Networks	torch.nn	Keras	Flax, Haiku
Gradient processing and optimization	torch.optim	tf.optimizers or tf.train	Optax
Checkpointing	torch.save, torch.load	tf.train.Checkpoint, tf.train.CheckpointManager	Orbax
Testing utilities	unittest, pytest	tf.test	Chex
Distributions	torch.distributions	TensorFlow Probability	Distrax, TensorFlow Probability
Parallelization	Not automatic	Not automatic	Automatic
Target Audience	Researcher, Developers	Researcher, Developers	Researchers
Maturity of Development	Mature	Mature	Not Mature

Table 2: Comparison of the three different ML Frameworks

http://www.imavs.org/

3.1 PyTorch

PyTorch is an open ML library developed by Meta AI Research, and now governed by PyTorch Foundation, a subsidiary of the Linux Foundation. PyTorch handles PyTorch tensors, similar to NumPy multidimensional arrays, but that can be operated on CUDA GPUs. Environments implemented using PyTorch tensors can be vectorized and hardware-accelerated, such as VMAS [13]. PyTorch benefits from a very consistent and mature ecosystem, with all main libraries needed for ML research being developed and maintained in the same package. It is an easily debuggable, hackable, and very flexible framework, and fits well into the Python ecosystem, allowing the use of Python debugger tools for debugging PyTorch code. It is targeted at both developers and researchers, who favor its Python-like imperative coding style, which makes it easy for developers to learn. Finally, PyTorch offers extensive community support and documentation and seamless deployment options.

3.2 TensorFlow

TensorFlow is an open ML library developed by Google. TensorFlow handles TensorFlow Tensors, similar to PyTorch ones, which can be operated on CUDA GPUs. Environments implemented using TensorFlow Tensors can be vectorized and hardware-accelerated, and TensorFlow provides an API as part of its RL library TF-Agents [22] to directly create parallelized and accelerated environments. However, it has yet to be broadly adopted. TensorFlow is known for its flexibility, scalability, and production readiness. TensorFlow also benefits from a very consistent ecosystem, with all main libraries needed for ML research being developed and maintained in the same package, except for neural network design, which is done with Keras' deep neural network API. However, we've seen a decline in the use of TensorFlow in both the industrial and research communities in the last few years, with researchers preferring the flexibility of PyTorch. Still, TensorFlow remains a dependable framework and offers extensive community support and documentation.

3.3 JAX

JAX is a Python library for accelerator-oriented array computation and program transformation, designed for high-performance numerical computing, built on top of Google's XLA (Accelerated Linear Algebra) library, and developed by researchers at Google and DeepMind. It's known for its automatic differentiation capabilities, which make it well-suited for optimization tasks such as training neural networks. JAX is also popular in the research community due to its auto-vectorization, device parallelism, and just-in-time compilation abilities, allowing to seamlessly parallelize Python code across multiple accelerators. JAX provides a NumPy-like interface, making it easy to transition from traditional numerical computing to more complex ML tasks. However, JAX is not an ML library, and its usage for machine RL requires dealing with various libraries, interoperable but independent [23, 24]. JAX is newer than PyTorch and TensorFlow and is celebrated for its immense scalability. However, it is at the edge of AI research and requires keeping pace with a rapidly evolving research ecosystem.

4 HARDWARE-ACCELERATED ENVIRONMENTS RECOMMENDED FEATURES

Here, we elaborate on the design choices and features required in environment implementation to accelerate the training pipeline in RL and enable their use for multi-drone applications. Table 3 summarizes key features and lack thereof in existing environments.

4.1 Vectorization

Vectorization is a technique in RL that enables parallel simulations. In a training iteration, the rollout phase is typically the most time-consuming part. Vectorization mitigates this by adding a batch dimension to the rollout phase, through sampling multiple sub-environments at the same time. Vectorization is something that is now widely adopted in RL,

with the most popular environment libraries such as Gymnasium [4] providing built-in vectorization.

4.2 Hardware acceleration

One straightforward method of vectorization is to simulate one environment instance per thread. However, this approach has limitations. The scalability is restricted by the number of physical cores in the system. Running a large number of threads can lead to synchronization issues, context-switching overhead, and memory bandwidth limitations. Moreover, the frequent switching between CPU cores, designed for sequential tasks, and GPUs, can be inefficient. It is possible to alleviate this, by leveraging the parallel processing and acceleration capabilities of GPUs.

GPU acceleration of the training environment is not something new. Gym-pybullet-drones [6] uses GPU-based rendering to speed up the simulation of multi-agent quadcopter control tasks. Moreover, a CUDA port of the Atari Learning Environment [26] was implemented by Dalton et al. However, these solutions still require data to be transferred back and forth between the CPU and the GPU when used in a training pipeline, introducing performance bottlenecks.

It is possible to eliminate those inefficiencies by keeping all the computations on the GPU [27]. GPU vectorization and parallelization allow seamless scaling to tens of thousands of parallel environments on accelerated hardware, speeding up training by order of magnitudes.

4.3 Python based

Python is an interpreted language, that is mostly known for its simplicity and rapidity of development rather than its execution performance. Complex environments implemented in Python require high CPU and memory requirements. The first attempts to alleviate this issue have been to implement the environments in C++. While it makes running the environments on the CPU much faster, it cannot be parallelized natively on the GPU, making it difficult to parallelize without large clusters of CPU. Currently, GPU Parallelization methods are typically implemented in CUDA C, such as [26]. However, the use of C++ and CUDA C considerably reduces their accessibility for most ML researchers who prefer Python [28]. Python environments enable a wider adoption in the MARL community.

New alternatives use Just-In-Time (JIT) compilation libraries, to compile Python code at execution. For example, WarpDrive [25] provides a Python interface and environment wrappers for implementing environments in CUDA C or Python using Numba, and automatically manages data transfer between the host and the device. Numba [29], optimizes Python code and allows easy Just-In-Time (JIT) compilation for Python array and numerical functions, significantly speeding up CPU code execution. It also supports CUDA GPU compilation for a limited subset of Python code, but this requires a solid understanding of the nuances and intricacies of the CUDA C language, as the majority of Numba GPU API functions map directly to Nvidia's CUDA C language. Tran-

Environment	Vector[a]	GPU[b]	Python[c]	PyTorch[d]	TensorFlow[e]	JAX[f]	Extens[g]	Open[h]	Compat[i]	Maint[j]
gymnasium [4]	✓	✗	✓	✗	✗	✗	✓	✓	✓	✓
PettingZoo [5]	✓	✗	✓	✗	✗	✗	✓	✓	✓	✓
Gym-Pybullet-drones [6]	✓	✗	✗	✗	✗	✗	✓	✓	✗	✓
EnvPool [7]	✓	✓	✗	✗	✗	✗	✓	✓	✓	✓
MAgent (original) [8]	✓	✓	✓	✗	✗	✗	✗	✓	✓	✗
brax [9]	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓
gymnax [10]	✓	✓	✗	✗	✗	✓	✗	✓	✓	✓
IsaacSIM [11]	✓	✓	✗	✓	✗	✗	✓	✗	✓	✓
Gigastep [12]	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓
VMAS [13]	✓	✓	✗	✓	✗	✗	✓	✓	✓	✓
JaxMARL [14]	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓
Jumanji [15]	✓	✓	✗	✗	✗	✓	✓	✓	✗	✓
CrazyRL (NumPy) [16]	✓	✗	✓	✗	✗	✗	✓	✓	✓	✗
CrazyRL (JAX) [16]	✓	✓	✗	✗	✗	✓	✓	✓	✓	✗
WarpDrive [25]	✓	✓	✗	✗	✗	✗	✗	✓	✗	✗

[a] Vectorization
[b] GPU acceleration and parallelization
[c] Python-based
[d] PyTorch-based
[e] TensorFlow-based
[f] JAX-based
[g] Extensible
[h] Open-Source
[i] Compatible with common APIs
[j] Maintained

Table 3: Comparison of different RL environments hardware-acceleration features

http://www.imavs.org/

http://www.imavs.org/

sitioning from CPU to GPU with Numba involves rewriting Python functions to manage kernel invocation and thread positioning, which is not straightforward.

As explained in Section 3, hardware acceleration in the field of machine learning in Python is instead typically done by adopting one of the three leading frameworks: PyTorch [1], TensorFlow [2], and JAX [3]. Vectorized environments with a batch dimension can be implemented using either PyTorch or TensorFlow tensors, allowing to parallelize and running them on GPU. However, the number of available PyTorch-based libraries is quite low, and no libraries were found using TensorFlow. An environment implemented using JAX arrays can be automatically vectorized to run a batch of this environment in parallel, and in comparison to the two other frameworks there have been more initiatives for parallelized environments using JAX presented in the literature, as JAX is very popular among researchers for its parallelization capabilities, and is seen as a successor of TensorFlow. However, there is limited cross-compatibility between these frameworks. Switching from one framework tensor representation to another requires moving the data on the CPU and inevitably introduces overheads in the training pipelines.

Each of these frameworks offers nowadays some JIT compilation functionality. Unlike Numba, JIT compilation with these frameworks for hardware accelerators doesn't require in-depth knowledge of low-level CUDA C details, and streamlines the process of adapting code for execution on a GPU automatically, simplifying the process. PyTorch offers limited JIT compilation with the TorchScript language, a typed subset of Python. It can be applied to any Python function that manipulates PyTorch tensors and supports PyTorch functions and many Python built-in functions. This is for example used in Isaac Gym [11], although the core functionalities are coded in C++ and CUDA C, a Python interface is provided for users to implement their own Python environment, and take advantage of TorchScript JIT to compile their Python functions to lower level scripts which are added to the parallelized training pipelines. TensorFlow allows using the XLA compiler through tf.function. It can be applied to any function that manipulates TensorFlow tensors, but it is not broadly used outside the optimization of neural network models and their training algorithms written with this framework [22]. In comparison, JAX JIT compilation using the XLA compiler is much more broadly adopted. It can be applied to any Python function that manipulates JAX arrays and automatically optimizes computations on hardware accelerators. One advantage over PyTorch is that JAX maintains most NumPy functions and their signatures, making it easier to directly adapt native Python code.

4.4 Compatible with common APIs

The Gym API [4], developed by OpenAI, has been a standard in the field since its release in 2016 and is the most used framework for coding simulators. DeepMind's environment API [23], known as dm_env, was introduced later in 2019 but is also broadly used. These APIs have played a significant role in shaping the development and standardization of RL environments. The majority of RL algorithm libraries, such as RLlib [30], Stable Baselines3 [31], and TorchRL [32], use these standardized APIs therefore, ensuring RL environments' compatibility with these standardized APIs is critical to ensure an easy integration in the RL ecosystem.

To enable batched execution of transition simulations, RL libraries are increasingly adopting *stateless* environments, where instead of environments keeping track of the last physical state encountered and relying on it to simulate the transition from state to state, stateless environments expect to be provided with the current state at each step, along with the action taken. With PyTorch, TensorFlow, or JAX, an algebraic operation can be seamlessly performed on scalars, vectors, or tensors, allowing the state to be easily batched to run multiple simulation steps in parallel. This is widely used in JAX because methods written this way are functionally pure and easily jittable. Gymnax [10] introduces such an API, where the Gym API is made stateless, and Jumanji [15] does a similar transformation for the dm_env API.

4.5 Open-source, Extensible, and Maintained

An environment library should be open-sourced and offer a high level of customization, to allow researchers and developers to inspect the code, ensuring that the algorithms and environments are implemented correctly and without hidden biases, to allow the research community to modify, adapt, or create custom scenarios that address their use cases, or to modify the code to specialize it for their application. Finally, as the RL research is rapidly evolving it is crucial that the library is still maintained, to ensure compatibility with the dependencies (in particular the last versions of Python) prevent issues that could arise from outdated code, and keep up with the advance and new features used in RL. Well-maintained code is also often a sign of trust within the community, showing that many users have adopted the library, are still contributing to it, and can provide active support.

5 TRAINING ON HARDWARE-ACCELERATED ENVIRONMENTS

Hardware-accelerated environments are increasingly gaining prominence in the single RL ecosystem. However, the existing RL libraries offering implementations of the state-of-the-art RL algorithm often fall short of effectively managing end-to-end training pipelines on accelerated hardware. This problem is further aggravated with MARL, as many of the most commonly used algorithm libraries lack native support for MARL. Table 4 summarizes how existing RL libraries support hardware-accelerated multi-agent environments.

One of the most popular RL libraries, RLlib [30], is implemented to facilitate highly distributed RL workloads. It can scale training from a single core to many thousands of

RL Libraries	Vector[a]	GPU[b]	Multi-GPU[c]	API[d]	MARL[e]	Audience[f]	PyTorch[g]	TensorFlow[h]	JAX[i]
Stable Baselines3 [31]	✓	limited	limited	gym	✗	D	✓	✗	limited
RLlib [30]	✓	✗	✗	gym	✓	D	✓	✓	✗
TorchRL [32]	✓	✓	✓	gym*	✓	R&D	✓	✗	limited
TF-Agents [22]	✓	✓	✗	dm_env	✗	D	✗	✓	✗
Acme [33]	✓	✓	✗	dm_env	✗	R	✗	✓	✓
CleanRL [34]	✓	with JAX	✗	any	limited	R	✓	✗	limited
PureJaxRL [35]	✓	✓	✗	gymnax*	✗	R	✗	✗	✓
JaxMARL [14]	✓	✓	✗	gymnax*	✓	R	✗	✗	✓
Mava [36]	✓	✓	✓	Jumanji	✓	R	✗	✗	✓

* provides wrappers for a limited number of additional APIs

[a] Handles vectorized environments
[b] Handles environments parallelized on GPU
[c] Handles environment parallelized on multiple GPUs
[d] Supported environment API
[e] Natively support for MARL
[f] Target Audience (R) Researchers (D) Developers
[g] PyTorch-based
[h] TensorFlow-based
[i] JAX-based

Table 4: Comparison of different RL libraries support for hardware-accelerated multi-agent environments

http://www.imavs.org/

cores in a cluster. Yet, it does not support end-to-end training pipelines on accelerated hardware as even if the environment was running on a hardware accelerator, the training pipeline requires the rollout data to be processed on the CPU, before sending it back to the GPU, where the agents are trained. These transfers back and forth introduce large overhead in the training pipeline.

Another difficulty is, the limited cross-compatibility between the three leading frameworks used for hardware acceleration: PyTorch [1], TensorFlow [2], and JAX [3]. Switching from one framework tensor representation to another requires moving the data on the CPU and inevitably introduces some overhead in the training pipeline. This means that if an RL environment library adopts one of these frameworks, it will face compatibility issues with RL algorithms that are designed for the other frameworks. This limits the mix-and-match options for RL environments and algorithms libraries.

The most popular RL libraries, such as Stable Baselines3 [31], are implemented in PyTorch or TensorFlow, which means that they do not support JAX-based environments natively, and require converting JAX arrays to their tensor representation to be able to process them. This is the case also with TorchRL [32], while it natively only supports end-to-end hardware-accelerated pipelines for PyTorch-based environments such as VMAS [32, 13], it provides a generic environment API and wrappers to support other existing simulators, such as Jumanji environments easily [15], which is JAX-based. However, JAX-based (and TensorFlow-based) environments inevitably suffer from a consequent overhead due to the conversion from JAX array to PyTorch Tensors.

	Single-Agent RL				MARL					
RL Libraries	DQL	DDPG	PPO	SAC	IPPO	MAPPO	IQL	MADDPG	QMIX	SAC
Stable Baselines3 [31]	FF - R	FF - R	FF - R	FF - R	-	-	-	-	-	-
RLlib [30]	FF - R	FF - R	FF - R	FF - R	FF - R	FF - R	FF - R	FF - R	-	FF - R
TorchRL [32]	FF	FF	FF	FF	FF	FF	FF - R	FF	FF	FF
TF-Agents [22]	FF - R	FF - R	FF - R	FF - R	-	-	-	-	-	-
Acme [33]	FF	FF	FF	FF	FF	-	-	-	-	-
CleanRL [34]	FF	FF	FF - R	FF	FF	FF	-	-	-	-
PureJaxRL [35]	FF	-	FF - R	-	-	-	-	-	-	-
JaxMARL [14]	-	-	-	-	FF - R	FF - R	R	-	R	-
Mava [36]	-	-	-	-	FF - R	FF - R	R	-	-	FF

FF — The library provides a Feed-Forward version of the baseline algorithm
R — The library provides a Recurrent version of the baseline algorithm

Table 5: Comparison of how RL libraries cover the existing state-of-the-art RL and MARL algorithms

http://www.imavs.org/

This lack of support for JAX-based environments by the most used and maintained RL libraries led the research community to create their own RL and MARL algorithms libraries implemented in JAX, such as PureJaxRL [35], JaxMARL [14] and Mava [36]. Often, while implementing and providing new environment libraries in JAX, the research community provides their own re-implementation of the baseline algorithms in JAX [14, 15, 36]. This results in an uneven offer of JAX implementations of the state-of-the-art RL algorithms. Often, these implementations lack certain training options or features, and their design choices are shaped by the benchmarked environments that accompany the algorithms. For example, the MAPPO implementation of Mava does not use value normalization, even though it is a recommended feature that has been shown to never hurt training and often improves the final performance of MAPPO significantly [20]. Again, this problem is worse with MARL algorithms. In particular, existing MARL libraries in JAX often lack the recurrent versions of the baselines algorithms, which are used to address partial observability in Dec-POMDPs, an intricate aspect of MARL problems such as the control of swarms of drones. The JAX ecosystem still lacks a unified, reliable, tested, and documented collection of RL algorithms. Table 5 provides a summary of how existing RL libraries cover the existing state-of-the-art RL and MARL algorithms.

6 CONCLUSION

The potential of vectorizable environments to leverage modern GPUs for parallel environment rollout promises a substantial boost in computational efficiency. However, the current landscape of RL algorithm libraries and frameworks falls short of effectively managing end-to-end training pipelines on accelerated hardware. Some initiatives have been introduced for parallelized and hardware-accelerated environments using PyTorch and JAX to a larger extent, while TensorFlow usage has been on the decline. However, the limited cross-compatibility between these frameworks hinders the seamless mix-and-match of RL environments and algorithm libraries, as switching from one framework tensor representation to another requires moving the data on the CPU and inevitably introduces some overhead in the training pipeline. This forces the research community to implement RL algorithm libraries for each framework, but while the most popular RL algorithm libraries are implemented in PyTorch and TensorFlow they have yet to fully adopt GPU-parallelized environments, and in contrast, the libraries implemented in JAX natively handle parallelized environments but lack maturity. A problem accentuated in MARL, as it is a specialized branch of RL research.

REFERENCES

[1] Adam Paszke, Sam Gross, Francisco Massa, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[2] Martín Abadi, Ashish Agarwal, Paul Barham, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[3] James Bradbury, Roy Frostig, Peter Hawkins, et al. JAX: composable transformations of Python+NumPy programs, 2018.

[4] Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, et al. Gymnasium, March 2023.

[5] J Terry, Benjamin Black, Nathaniel Grammel, et al. Pettingzoo: Gym for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:15032–15043, 2021.

[6] Jacopo Panerati, Hehui Zheng, SiQi Zhou, et al. Learning to Fly—a Gym Environment with PyBullet Physics for Reinforcement Learning of Multi-agent Quadcopter Control. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7512–7519, September 2021.

[7] Jiayi Weng, Min Lin, Shengyi Huang, et al. EnvPool: A Highly Parallel Reinforcement Learning Environment Execution Engine. *Advances in Neural Information Processing Systems*, 35:22409–22421, December 2022.

[8] Lianmin Zheng, Jiacheng Yang, Han Cai, et al. MAgent: A Many-Agent Reinforcement Learning Platform for Artificial Collective Intelligence. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), April 2018.

[9] C. Daniel Freeman, Erik Frey, Anton Raichuk, et al. Brax – a differentiable physics engine for large scale rigid body simulation, June 2021.

[10] Robert Tjarko Lange. gymnax: A JAX-based reinforcement learning environment library, 2022.

[11] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, et al. Isaac gym: High performance gpu-based physics simulation for robot learning, August 2021.

[12] Mathias Lechner, Lianhao Yin, Tim Seyde, et al. Gigastep - One Billion Steps per Second Multi-agent Reinforcement Learning. *Advances in Neural Information Processing Systems*, 36:155–170, December 2023.

[13] Matteo Bettini, Ryan Kortvelesy, Jan Blumenkamp, and Amanda Prorok. VMAS: A Vectorized Multi-agent Simulator for Collective Robot Learning. In Julien Bourgeois, Jamie Paik, Benoît Piranda, et al., editors, *Distributed Autonomous Robotic Systems*, pages 42–56, Cham, 2024. Springer Nature Switzerland.

[14] Alexander Rutherford, Benjamin Ellis, Matteo Gallici, et al. Jaxmarl: Multi-agent rl environments in jax, December 2023.

[15] Clément Bonnet, Daniel Luo, Donal Byrne, et al. Jumanji: a diverse suite of scalable reinforcement learning environments in jax, March 2023.

[16] Florian Felten, Coline Ledez, Pierre-Yves Houitte, et al. Crazyrl: A multi-agent reinforcement learning library for flying crazyflie drones. https://github.com/ffelten/CrazyRL, 2023.

[17] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, et al. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. pages 4295–4304. PMLR, July 2018.

[18] Jianyu Su, Stephen Adams, and Peter Beling. Value-Decomposition Multi-Agent Actor-Critics. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13):11352–11360, May 2021.

[19] Ryan Lowe, YI WU, Aviv Tamar, et al. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[20] Chao Yu, Akash Velu, Eugene Vinitsky, et al. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. *Advances in Neural Information Processing Systems*, 35:24611–24624, December 2022.

[21] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. pages 1861–1870. PMLR, July 2018.

[22] Sergio Guadarrama, Anoop Korattikara, Oscar Ramirez, et al. TF-Agents: A library for reinforcement learning in tensorflow. https://github.com/tensorflow/agents, 2018.

[23] DeepMind, Igor Babuschkin, Kate Baumli, et al. The DeepMind JAX Ecosystem, 2020.

[24] Jonathan Heek, Anselm Levskaya, Avital Oliver, et al. Flax: A neural network library and ecosystem for JAX, 2023.

[25] Tian Lan, Sunil Srinivasa, Huan Wang, and Stephan Zheng. WarpDrive: Extremely Fast End-to-End Deep Multi-Agent Reinforcement Learning on a GPU, October 2021.

[26] Steven Dalton and iuri frosio. Accelerating Reinforcement Learning through GPU Atari Emulation. In *Advances in Neural Information Processing Systems*, volume 33, pages 19773–19782. Curran Associates, Inc., 2020.

[27] Matteo Hessel, Manuel Kroiss, Aidan Clark, et al. Podracer architectures for scalable reinforcement learning. April 2021.

[28] Sebastian Raschka, Joshua Patterson, and Corey Nolet. Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence. *Information*, 11(4):193, April 2020.

[29] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: a LLVM-based Python JIT compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, LLVM '15, pages 1–6, New York, NY, USA, November 2015. Association for Computing Machinery.

[30] Eric Liang, Richard Liaw, Robert Nishihara, et al. RLlib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.

[31] Antonin Raffin, Ashley Hill, Adam Gleave, et al. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[32] Albert Bou, Matteo Bettini, Sebastian Dittert, et al. Torchrl: A data-driven decision-making library for pytorch, November 2023.

[33] Matthew W. Hoffman, Bobak Shahriari, John Aslanides, et al. Acme: A research framework for distributed reinforcement learning, September 2020.

[34] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, et al. CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

[35] Chris Lu, Jakub Kuba, Alistair Letcher, et al. Discovered Policy Optimisation. *Advances in Neural Information Processing Systems*, 35:16455–16468, December 2022.

[36] Ruan de Kock, Omayma Mahjoub, Sasha Abramowitz, et al. Mava: a research library for distributed multi-agent reinforcement learning in jax, December 2021.

http://www.imavs.org/

Adaptive Neural Network Quadrotor Trajectory Tracking Controller Tolerant to Propeller Damage

Mauro Villanueva Aguado*, Christophe De Wagter†, and Guido de Croon‡
Micro Air Vehicle Lab, Faculty of Aerospace Engineering,
Delft University of Technology, 2629 HS Delft, The Netherlands

http://www.imavs.org/

ABSTRACT

Accurate trajectory tracking with quadrotors is a challenging task that requires a trade-off between accuracy and complexity to run onboard. State-of-the-art adaptive controllers achieve impressive trajectory tracking results with slight performance degradation in varying winds or payloads but at the cost of computational complexity. This work proposes a lightweight combination of adaptive and neural control and shows its performance when flying with propeller damage. The neural architecture consists of offline learning of a condition-invariant representation of the aerodynamic forces through Deep Neural Networks. The second part consists of fast online adaptation using a composite adaptation law. We deploy this flight controller fully onboard the flight controller of the Parrot Bebop 1, showcasing its computational efficiency. The adaptive neural controller improves tracking performance by $\approx$ 60% over the nonlinear baseline, with minimal performance degradation of just $\approx$ 20% with increasing propeller damage.

1 INTRODUCTION

Unmanned Aerial Vehicles (UAVs) offer unmatched versatility & agility, with the potential to revolutionize a wide range of industries including cinematography, defense, agriculture, and logistics. Crucially UAVs require a trained operator to fly, increasing operational costs, and hindering the wider commoditization of UAVs. Autonomous UAVs take away the need for a trained operator. Autonomous UAVs have already been deployed in niche applications, from making history by executing the first controlled flight on another planet [1] to delivering crucial medical supplies to isolated hospitals in Africa [2].

One of these challenges is tracking trajectories accurately and thus safely in a wide range of conditions, which relies on aerodynamics modeling of the quadrotor. But these aerodynamic effects are difficult to model, "they consist of the propeller lift and drag which are dependent on the induced air stream velocity, fuselage drag, downwash and turbulent interactions between propellers and fuselage" [3]. This problem is further compounded in the case that the propellers are damaged which significantly changes the aerodynamic model of the quadrotor.

Conventional approaches to trajectory tracking use simple linear or quadratic [4] drag models to capture these aerodynamic effects. While simple yet effective at low speeds the accuracy of these models quickly degrades at higher speeds and cannot account for changes in the aerodynamic model. While high-fidelity aerodynamic models have been derived from Computational Fluid Dynamic (CFD) simulations [5], these simulations require platform-specific meshing of hundreds of millions of grid points and multiple days in large compute clusters to solve and are ultimately condensed into simplified models to be tractable onboard. Advances in Neural Networks (NN) and small powerful GPUs have enabled promising results at approximating these aerodynamic effects [6, 7, 8] while remaining computationally tractable to run onboard a quadrotor. NN provide a higher accuracy than the simple drag models. However, implementing NN in a drone controller while addressing the unpredictable nature of their output remains a largely unexplored challenge. A trade-off between model accuracy & complexity must be considered.

This paper is structured as follows. An overview of related work on quadrotor trajectory tracking control, adaptive control and adaptive neural control is presented in section 2. In section 3 a detailed methodology of the quadrotor model, adaptive neural controller and trajectory generation is provided. The experimental setup is explained in section 4. The main results are shown in section 5, including neural network training & validation loss, unmodeled force predictions, controller tracking performance comparison and trajectory plots. This is followed by discussing the results and providing recommendations in section 6. Lastly, the main conclusions of the adaptive neural controller are raised in section 7.

2 RELATED WORK

2.1 *Quadrotor Trajectory Tracking Control*

Quadrotors are inherently nonlinear and unstable platforms. Early work on quadrotor control achieved stable hover and near-hover flight using well-established control schemes such as Proportional-Integral-Derivative (PID) [9] or Linear-Quadratic Regulator (LQR) [10]. These controllers rely on the small-angle assumption to linearize the dynamics of the

*Email address: contact@maurovillanueva.eu
†Email address: c.dewagter@tudelft.nl
‡Email address: g.c.h.e.decroon@tudelft.nl

http://www.imavs.org/

system, thus they are most effective at low speeds. Despite this drawback cascaded PID control is the most common controller architecture in off-the-shelf flight controllers due to its simplicity, ease-of-tuning and sufficiently good performance.

As quadrotors have become more powerful and lightweight, capable of executing aggressive trajectories, nonlinear flight controllers have been proposed to cope with the non-linearities in the attitude dynamics. Nonlinear flight controllers such as Nonlinear Dynamic Inversion (NDI) [11] transform the nonlinear dynamics into a linear input-output map enabling the use of a linear control law. However, exact dynamic inversion suffers from a lack of robustness as it quite sensitive to sensor noise as well as modeling uncertainty [12]. Variants of NDI have been proposed to solve this problem, such as backstepping design [13] which recursively designs a controller, starting from a known stable inner system and progressively *backs-out* stabilizing the entire nonlinear system. A more recent variant known as incremental nonlinear dynamic inversion (INDI) [14] has been developed, which improves robustness by gradually applying control inputs based on inertial measurements.

State-of-the-art trajectory tracking algorithms can be categorized into predictive or non-predictive methods. Non-predictive methods track a single reference step while predictive methods encode several future timesteps into the control command. Differential-Flatness-Based Control (DFBC), a non-predictive method, takes advantage of the fact that quadrotors are differentially flat systems [15] allowing for the reformulation of the trajectory tracking problem into a state tracking problem. DFBCs are combined with an outer loop INDI controller and aerodynamic model to achieve an impressive tracking performance at high-speeds [16, 17], with a Position Root-Mean-Squared Error (RMSE) of as little as $12.2\ cm$ with a top speed of $20\ m/s$ [17]. However, [17] relies on optical encoders for direct motor speed feedback or in the case [16], is limited to x-y plane trajectories and uses a very high-frequency attitude controller running at $4\ kHz$. Most importantly, DFBC relies on the quadrotor model to generate the reference states and thus is susceptible to modeling mismatch, reducing the thrust coefficient by 30% will lead to significant performance degradation or even crashing, with position RMSE degrading from $8.5\ cm$ to $100.4\ cm$ [18].

Nonlinear Model Predictive Control (NMPC) is the most prevalent predictive method. NMPCs generate motor commands in a receding horizon fashion, solving the constrained nonlinear optimization problem over the predicted time horizon and minimizing tracking error. Similarly, the performance of NMPCs is further improved by adding an outer loop INDI controller and aerodynamic model. NMPCs can minimize the tracking error across multiple future time-steps whereas DFBCs are too short-sighted only considering one reference point. This allows NMPCs to outperform DFBCs at tracking trajectories at high speeds, especially for dynamically infeasible trajectories. NMPCs achieve a state-of-the-art position RMSE of $10.2\ cm$ with a maximum speed of $20\ m/s$ [18]. However, solving this nonlinear optimization problem over several future time steps requires significant computational resources, in the order of 100 times greater than DFBC. Despite advances in powerful embedded computers and nonlinear solvers deploying NMPC onboard is still a challenge, requiring an NVIDIA Jetson TX2 to run the algorithm onboard at $100\ Hz$ in [18] adding significant weight and power consumption.

2.2 *Adaptive Control*

Adaptive control of systems with parametric uncertainty has been extensively researched. Adaptive controllers are often an augmentation to existing controllers rather than a standalone controller. In the field of UAVs adaptive controllers can be categorized into Model Identification Adaptive Controllers (MIACs) or Model Reference Adaptive Controllers (MRACs). MRACs use an adaptive controller, typically an $\mathcal{L}_1$ adaptive controller, to drive the system towards a desired reference model behavior. MRACs have been successfully deployed in quadrotors, demonstrating slight trajectory tracking degradation with unmodeled weights attached [19, 20] or loss-of-thrust [21]. Albeit, these results are achieved tracking simple circular trajectories not exceeding $2\ m/s$ with a ground station in-the-loop [19], require powerful embedded computers [20] or have overall poor tracking performance [21].

MIACs perform System Identification (SI) online to estimate the values of unknown linear coefficients which are then mixed with known basis functions. The selection of basis functions may be challenging, but they should reflect important features of the underlying dynamics. Physics-based modeling is typically used to design these basis functions [22], however, this requires extensive knowledge of the system and suffers from convergence problems when there is a lack of excitation. Alternatively, random Fourier features can be used as basis functions as they can capture all underlying dynamics given a sufficient number of features. However, the high-dimensional feature space may not be an efficient representation as features are redundant or irrelevant and lack the representation power of Deep Neural Networks (DNNs). Nevertheless, random Fourier features are used in [23] to learn an acceleration error model of a quadrotor with a drag plate attached in windy conditions improving on the performance of an $\mathbf{L}_1$ adaptive controller.

2.3 *Adaptive Neural Control*

The ability of NNs to accurately and computationally efficiently approximate nonlinear functions has drawn interest in the field of adaptive control. Early work focused on solving the identification problem of nonlinear systems using NNs to create a reference model [24, 25] whose weights are updated online based on the error between the system and model output. These networks were shallow and lacked pretraining, limiting their performance and requiring the initial guess of

http://www.imavs.org/

network weights to be close to the correct guess to ensure stability. More recently, a quadrotor has performed an impressive flip in wind using an NN to predict the unknown forces [7], the weights are updated based on the position & velocity error. Albeit, the network is still limited to a simple 3-layer [6,3,3] neuron architecture lacking pretraining.

Lately, meta-learning has drawn attention as a scheme that "learns-to-learn" efficient models of systems from data gathered in varying conditions. The learned model is capable of fast adaptation to highly dynamic environments. Meta-Learning algorithms typically can be decomposed into two phases: offline learning and online adaptation. In the offline learning phase, the goal is to learn a model from data collected in various environments that capture common features shared across all environments. In the online adaptation phase, the goal is to adapt this model to the current environment based on the data available. A naive approach to online adaptation would adapt the entire network online, however this is a computationally expensive task severely limiting the depth of the network and lacks stability guarantees to unpredictable outputs. A more sophisticated approach augments the NN output with latent variables identified online which represent the unknown environmental factors allowing for rapid adaptation with little computational burden. Meta-Learning adaptive control has achieved successful adaptive control of quadrotors carrying suspended payloads [26] and strong winds [27]. Of particular interest are the results achieved in [27] whose "Neural-Fly" controller demonstrates state-of-the-art tracking performance in strong & varying wind conditions achieving a position RMSE of $9.4cm$ at wind speeds of $12.1\ m/s$ with little computational cost.

3 METHODOLOGY

In this paper vectors are denoted in lowercase bold $\mathbf{p}$ and matrices in uppercase bold $\mathbf{I}$, otherwise, they are scalars. Two right-handed coordinate frames are used in this paper shown in Figure 1, the body frame: $\mathcal{F}_B : \{\boldsymbol{x}_B, \boldsymbol{y}_B, \boldsymbol{z}_B\}$ located at the center of mass of the quadrotor with $\boldsymbol{x}_B$ pointing forward and $\boldsymbol{z}_B$ aligned with the collective thrust direction and the inertial world frame: $\mathcal{F}_W : \{\boldsymbol{x}_W, \boldsymbol{y}_W, \boldsymbol{z}_W\}$ with $\boldsymbol{z}_W$ pointing in the opposite direction of gravity. Vectors with the superscript $\square^B$ are expressed in the body frame and those without one are expressed in the world frame. The rotation from inertial frame to body frame is represented by the rotational matrix $\boldsymbol{R} = [\boldsymbol{x}_B, \boldsymbol{y}_B, \boldsymbol{z}_B] \in \mathrm{SO}(3)$.

3.1 Quadrotor Model

The quadrotor model is based on the model provided in [17], modeled as a 6-degree-of-freedom rigid body with 4 inputs corresponding to the commanded rpms of the motors. The translational dynamics is given by:

$$\boldsymbol{a}^I = \frac{\boldsymbol{f}_t + \boldsymbol{f}_a + \boldsymbol{f}_{res}}{m} + \boldsymbol{g} \tag{1}$$

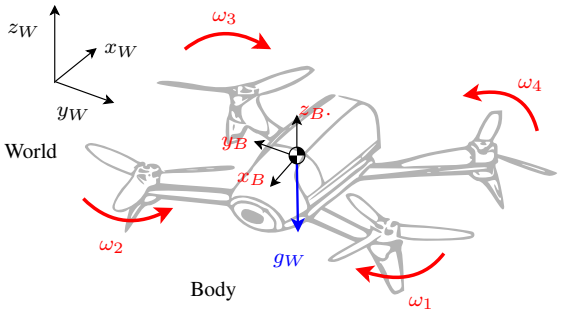


Figure 1: Coordinate System Definitions

where $\boldsymbol{a}$ is the acceleration of the quadrotor's center of gravity; $\boldsymbol{f}_t$ is the collective thrust; m is the quadrotor mass; $\boldsymbol{f}_a$ represents the aerodynamic model forces; $\boldsymbol{f}_{res}$ captures the remaining unmodeled forces; $\boldsymbol{g} = \left[0, 0, -9.81\ \mathrm{ms}^{-2}\right]^T$ is the gravity vector. The rotational kinematics and dynamic equations are expressed as:

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \otimes \boldsymbol{\Omega}^B \tag{2}$$

$$\dot{\boldsymbol{\Omega}}^B = \boldsymbol{I}^{-1}\left(\boldsymbol{\tau}^B + \boldsymbol{\tau}^B_{ext} - \boldsymbol{\Omega}^B \times \boldsymbol{I}\boldsymbol{\Omega}^B\right) \tag{3}$$

where $\boldsymbol{\Omega}^B$ is the body frame angular velocity vector; $\boldsymbol{I} \in \mathbb{R}^{3\times3}$ is the quadrotor diagonal moment of inertia matrix; $\boldsymbol{\tau}$ are the modeled body torques and $\boldsymbol{\tau}_{ext}$ accounts for the any unmodeled disturbance moments. The symbol $\otimes$ represents the Hamilton quaternion product.

3.1.1 Thrust and Torque Model

The thrust and torque acting on the quadrotor are modeled in Equation 4 & Equation 5 respectively, we assume stiff propellers with no rotor drag and neglect moments caused by aerodynamic effects, the angular acceleration of rotors and gyroscopic effects.

$$\boldsymbol{f}_t^B = \left[0, 0, c_t \sum_{i=1}^{4} \omega_i^2\right]^T \tag{4}$$

$$\boldsymbol{\tau}^B = \begin{bmatrix} l_y c_t & -l_y c_t & -l_y c_t & l_y c_t \\ l_x c_t & l_x c_t & -l_x c_t & -l_x c_t \\ c_\tau & -c_\tau & c_\tau & -c_\tau \end{bmatrix} \boldsymbol{\omega}^2 \tag{5}$$

where c_t is the propeller thrust coefficient and c_τ the propeller torque coefficient; l_x and l_y are the moment arms shown in Figure 1.

3.1.2 Aerodynamic Force Model

The aerodynamic model from [4] captures the major aerodynamic effects in a computationally efficient manner.

$$\boldsymbol{f}_a^B = -\sum_{i=1}^{4} \omega_i \begin{bmatrix} k_x \\ k_y \\ k_z \end{bmatrix} \odot \boldsymbol{v}^B + \begin{bmatrix} 0 \\ 0 \\ k_h \left({v_x^B}^2 + {v_y^B}^2 \right) \end{bmatrix} \quad (6)$$

where $\boldsymbol{v}^B = \boldsymbol{R}(\boldsymbol{q})^T \boldsymbol{v}$ is the body frame velocity vector and k_x, k_y, k_z, k_h are the aerodynamic coefficients identified from flight data in [28]. The symbol $\odot$ represents vector element-wise multiplication.

3.2 Adaptive Neural Controller

The adaptive neural controller design follows the "Neural-Fly" controller architecture closely in [27]. Neural-Fly is a data-driven trajectory tracking controller that uses a learning-based approach to achieve fast & accurate online adaption by incorporating pre-trained representations using deep learning. The controller has two main components: offline meta-learning and online adaptive control. These two components together build a model of the unmodeled forces acting on a quadrotor of the form:

$$\boldsymbol{f_{res}}(\boldsymbol{x}, \boldsymbol{k}) \approx \phi(\boldsymbol{x}) \boldsymbol{A}(\boldsymbol{k}) \quad (7)$$

where $\boldsymbol{\phi}$ is a representation function that maps the unmodeled forces dependent on the quadrotors state $\boldsymbol{x}$ to a shared representation across all sampled environments. $\boldsymbol{A}$ is a set of linear coefficients that adapt to the current quadrotor environment k. The offline training phase consists of training $\boldsymbol{\phi}$, a DNN using Domain Adversarial Meta-Learning (DAIML) to learn a propeller damage invariant representation of the unmodeled forces. The online adaptation phase aims to adapt the linear coefficients in $\boldsymbol{A}$ to the current propeller damage condition using a composite adaptive controller while maintaining stability and robustness.

Combining the translational dynamics of a quadrotor in Equation 1 with the estimate of the unmodeled force in Equation 7 we arrive at the control law of the adaptive neural controller:

$$\boldsymbol{u}_f = \underbrace{m\boldsymbol{a}_r^B + \boldsymbol{R}\boldsymbol{f}_a^B + m\boldsymbol{g}^B}_{\text{nominal model feedforward}} \underbrace{-\boldsymbol{K}_p\boldsymbol{e}_p - \boldsymbol{K}_v\boldsymbol{e}_v}_{\text{PD feedback}} \underbrace{-\boldsymbol{\phi}(\boldsymbol{x})\hat{\boldsymbol{A}}}_{\text{adaptation}} \quad (8)$$

where $\boldsymbol{e}_p = p - p_r$ is the position tracking error, the subscript $\square_r$ denotes the reference trajectory, i.e. a_r is the reference acceleration of the desired trajectory; $\boldsymbol{e}_v = v - v_r$ is the velocity tracking term; $\boldsymbol{K}_p$ and $\boldsymbol{K}_v$ are positive definite control gain matrices for position and velocity respectively. The output of Equation 8 is the control law u_f, a vector of commanded forces from which the corresponding attitude and thrust is obtained through inverse kinematics.

The adaptive neural controller replaces the position control loop, and the output attitude and thrust commands are sent to the inner attitude control loop and thrust mixer respectively. In contrast to a traditional PID controller, the adaptive neural controller also includes nominal model feed-forward terms to account for the known aerodynamic effects and desired acceleration, improving tracking of the reference trajectory. The architecture of the adaptive neural controller is shown in Figure 2.

This paper builds upon the work presented in [27] through the following contributions: (I) validation of the "Neural-Fly" adaptive neural controller architecture (II) adaptation to novel propeller damage conditions (III) analysis of the unmodeled force prediction performance on training, testing and flight data (IV) sensitivity analysis of the neural network architecture to hyperparameters (VI) implementation of the online adaptation phase onboard the flight controller in C.

3.2.1 Offline Training

The goal of the offline training phase is to learn a representation function $\boldsymbol{\phi}(\boldsymbol{x})$ such that for any condition k a latent variable $\boldsymbol{A}$ exists such that $\boldsymbol{\phi}(\boldsymbol{x})\boldsymbol{A}(k)$ approximates $f_{res}(\boldsymbol{x}, k)$ well. A DNN is used to learn this representation function $\boldsymbol{\phi}$, taking advantage of the representation power of DNNs to accurately approximate any nonlinear function given a sufficient amount of neurons and training data. The optimal representation $\boldsymbol{\phi}^*$ is typically obtained by minimizing the loss:

$$\min_{\boldsymbol{\phi}, \boldsymbol{A}^1, \cdots, \boldsymbol{A}^K} \sum_{k=1}^{K} \sum_{i=1}^{N^k} \left\| \boldsymbol{y}_i^k - \boldsymbol{\phi}\left(\boldsymbol{x}_i^k\right) \boldsymbol{A}^k \right\| \quad (9)$$

where K is the total amount of conditions sampled and N_k is the total amount of data-points collected with condition k. It is the Euclidean norm of the error between the measured unmodeled forces $\boldsymbol{y}$ and the network output $\boldsymbol{\phi}(x)\boldsymbol{A}$ across all sampled conditions and data points.

Note that the minimization problem also involves the latent variables $\boldsymbol{A}^1, \cdots, \boldsymbol{A}^K$, thus back-propagation is also performed through $\boldsymbol{A}$ to ensure that the optimal latent variables $\boldsymbol{A}^{k^*}$ are found.

However, the distribution of the quadrotor states $\boldsymbol{x}$ will vary depending on the propeller damage condition k flown. For example, propeller rpm values will increase as propeller damage worsens to cope with the loss of lift. This inherent domain shift in x caused by a shift in k may lead to overfitting of the network $\boldsymbol{\phi}$. The network $\boldsymbol{\phi}$ could learn the change in the unmodeled forces due to k via the change in the distribution of x, instead of learning a propeller damage invariant representation. To solve this domain shift problem an adversarial loss is used:

$$\max_{h} \min_{\boldsymbol{\phi}, \boldsymbol{A}^1, \cdots, \boldsymbol{A}^K} \sum_{k=1}^{K} \sum_{i=1}^{N^K} \left(\left\| \boldsymbol{y}_i^k - \boldsymbol{\phi}\left(\boldsymbol{x}_i^k\right) \boldsymbol{A}^k \right\| - \alpha \cdot \mathcal{L}_{\boldsymbol{h}}\left(\boldsymbol{h}\left(\boldsymbol{\phi}\left(\boldsymbol{x}_i^k\right)\right), k\right) \right) \quad (10)$$

http://www.imavs.org/

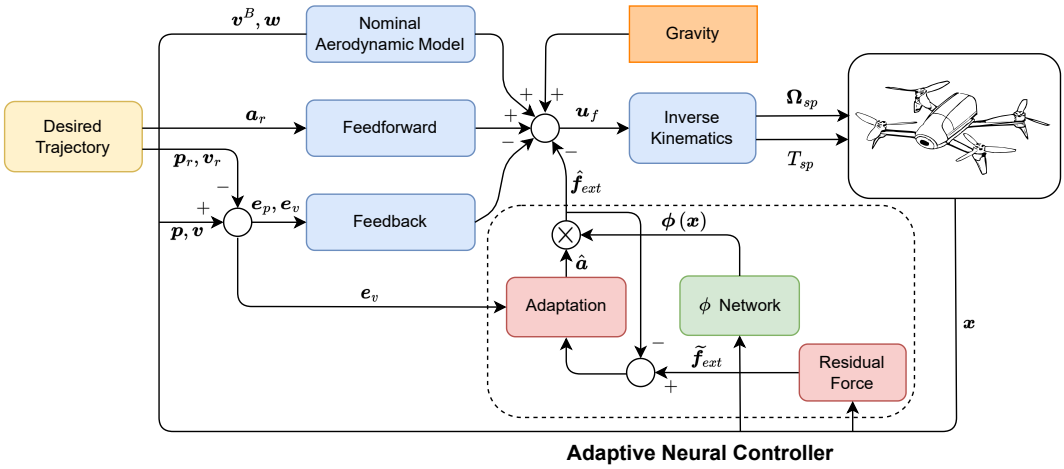


Figure 2: Control Diagram of the Adaptive Neural Controller

In this adversarial loss the discriminator network $\boldsymbol{h}$ is competing against $\boldsymbol{\phi}$ in a zero-sum game. The network $\boldsymbol{h}$ attempts to predict the condition k from the output of the network $\boldsymbol{\phi}(\boldsymbol{x})$, while $\boldsymbol{\phi}$ attempts to approximate the residual force measurement y while making the job of h harder. The influence of the discriminator network h is controlled through the regularization term $\alpha \geq 0$. $\mathcal{L}_{\boldsymbol{h}}$ is the loss function of h, typically a cross-entropy loss function is used to train classification models:

$$\mathcal{L}_{\boldsymbol{h}} = -\sum_{i \in B} \delta_{k=j} \odot \log\left(\boldsymbol{h}\left(\boldsymbol{\phi}\left(x_i^k\right)\right)\right) \tag{11}$$

The cross-entropy loss compares the log of the output of the discriminator network $\boldsymbol{h}\left(\boldsymbol{\phi}\left(x_i^k\right)\right)$, a vector containing the classification probabilities of each of the K conditions, with the ground truth classification vector δ_i. For example, if a data point is collected with no propeller damage with index $k = 1$ the corresponding ground truth vector is $\delta_i = [1, 0, 0]$.

The DAIML algorithm developed in [27] is shown in algorithm 1. DAIML consists of three main steps: (I) **the adaptation step** solves the least squares problem on the adaptation batch B^a to find the optimal latent variable $\boldsymbol{A}^*$ with a fixed network $\boldsymbol{\phi}$, (II) **the $\boldsymbol{\phi}$ training step** updates the parameters of the network $\boldsymbol{\phi}$ using Stochastic Gradient Descent (SGD) based on the adversarial loss over the training batch $\boldsymbol{B}^t$ with $\boldsymbol{A}^*$, (III) **the $\boldsymbol{h}$ training step** updates the parameters of the discriminator network h using SGD based on the cross entropy loss over the training batch $\boldsymbol{B}^t$.

Nevertheless, training adversarial networks is a challenging task due to problems such as mode collapse, vanishing gradients and unstable convergence. Fortunately, Generative Adversarial Networks (GANs) have been extensively researched and various well-documented features have been shown to improve training. These improvements have been implemented in DAIML they include (1) normalization of the latent variable $||\boldsymbol{A}^*|| > \gamma$ to improve robustness, (2) spectral normalization of the weights of the discriminator network $\boldsymbol{W}_h = \boldsymbol{W}_h / \sigma(\boldsymbol{W}_h)$ to improve the stability, (3) training the networks $\boldsymbol{\phi}$ and $\boldsymbol{h}$ in an alternating manner as well as training the discriminator less frequently to improve convergence.

Algorithm 1: DAIML

Hyperparameters: $\alpha \geq 0,\ \gamma > 0,\ 0 < \eta \leq 1$
Data: $\boldsymbol{D} = \{\boldsymbol{D}^1, \ldots, \boldsymbol{D}^K\}$
Result: trained neural network $\boldsymbol{\phi}$ and $\boldsymbol{h}$

while *not converged* **do**
  Randomly sample $\boldsymbol{D}^k \in \boldsymbol{D}$
  Randomly sample disjoint batches $\boldsymbol{B}^a, \boldsymbol{B}^t \in \boldsymbol{D}^k$
  Solve $\boldsymbol{A}^*(\phi) = \arg\min_{\boldsymbol{A}} \sum_{i \in B^a} \left\| y_i^k - \boldsymbol{\phi}\left(x_i^k\right) \boldsymbol{A} \right\|$
  Train $\boldsymbol{\phi}$ with loss: $\mathcal{L}_{\boldsymbol{\phi}} = \sum_{i \in B^t} \left(\left\| y_i^k - \boldsymbol{\phi}\left(x_i^k\right) \boldsymbol{A}^* \right\| - \alpha \cdot \mathcal{L}_{\boldsymbol{h}}\left(\boldsymbol{h}\left(\boldsymbol{\phi}\left(x_i^k\right)\right), k\right)\right)$
  if $X \sim U(0,1) \leq \eta$ **then**
    Train $\boldsymbol{h}$ with cross-entropy loss: $\mathcal{L}_{\boldsymbol{h}} = -\sum_{i \in B^t} \delta_i \odot \log\left(\boldsymbol{h}\left(\boldsymbol{\phi}\left(x_i^k\right)\right)\right)$
  end
end

3.2.2 Online Adaptation

An online adaptation law based on a Kalman-filter estimator is used to update the estimate of the linear coefficients $\hat{\boldsymbol{A}}$ through Equation 12. While in the offline training phase, the optimal coefficients $\boldsymbol{A}^*$ are obtained by minimizing the least squares force prediction error. In the online adaptation phase, the goal is to minimize the position tracking error. Thus, the linear coefficients are updated based on a composite er-

http://www.imavs.org/

ror, consisting of the force prediction error and the tracking error, improving the trajectory tracking performance of the controller. Additionally, the Kalman-filter estimator automatically tunes the gain matrix P given by Equation 13. This allows the online adaptation law to estimate the complex latent variable in the presence of varying uncertainties, alleviating the need for constant excitation for accurate estimation.

$$\dot{\hat{\boldsymbol{A}}} = \underbrace{-\lambda \hat{\boldsymbol{A}}}_{\text{regularization}} \underbrace{- \boldsymbol{P}\phi(\boldsymbol{x})^{\top} \boldsymbol{R}^{-1}(\phi(\boldsymbol{x})\hat{\boldsymbol{A}} - \boldsymbol{y})}_{\text{prediction error}} + \underbrace{\boldsymbol{P}\phi(\boldsymbol{x})^{\top} \boldsymbol{s}}_{\text{tracking error}} \tag{12}$$

$$\dot{\boldsymbol{P}} = -2\lambda \boldsymbol{P} + \boldsymbol{Q} - \boldsymbol{P}\phi(\boldsymbol{x})^{\top} \boldsymbol{R}^{-1} + \phi(\boldsymbol{x})\boldsymbol{P} \tag{13}$$

In practice, the adaptation law is implemented in a digital system which requires the discrete version of the Kalman filter. This discrete Kalman filter suffers from numerical stability and convergence issues, particularly in the covariance matrix P. As discussed in [27], coarse integration step sizes of the continuous time adaptation law would sometimes result in P becoming non-positive-definite. Their proposed solution, splitting the adaptation law into two steps: a propagation step and an update step is implemented.

3.3 *Trajectory Generation*

The desired trajectory of the quadrotor is defined using a spline that minimizes snap, the 4th derivative of position with respect to time $\ddddot{\boldsymbol{p}}$. Minimum snap trajectories are desirable for quadrotor tracking since motor commands and attitude accelerations are proportional to snap. Moreover, polynomial splines are easily differentiable and useful when calculating the velocity and acceleration of the reference trajectory.

Given a list of N waypoints $\boldsymbol{W} = [\boldsymbol{W}_1, \ldots, \boldsymbol{W}_N]$ and $W_n \in \mathbb{R}^3$ the desired position of the quadrotor is given by $\boldsymbol{p}_m = \left[p_{m_x}(t), p_{m_y}(t), p_{m_z}(t)\right]$ with each axis described by one spline consisting of $M = N - 1$ polynomials. The minimum snap spline of each axis is obtained by solving the minimization problem:

$$\min_{p_1,\cdots,p_M} \sum_{m=1}^{M} \int_0^T \| \ddddot{p}_m(t) \| \, dt \tag{14}$$

where T is the amount of time for each waypoint segment. Given that the distance between waypoints is arbitrary using the same amount of time for all segments severely constrains the solution quality. To improve the overall solution the segment times are included in the cost function in Equation 15 and allowed to vary. However, changing the segment times also changes the cost function resulting in a non-convex optimization problem, an initial guess of segment times is required and then iteratively refined using gradient descent to reach a minimum.

$$\min_{p_1,T_1,\cdots,p_M,T_M} \sum_{m=1}^{M} \int_0^{T_m} \| \ddddot{p}_m(t) \| \, dt + \mu T_m \tag{15}$$

Subject to:

$$\boldsymbol{p}_m(0) = \boldsymbol{W}_m, \quad \boldsymbol{p}_m(T_m) = \boldsymbol{W}_{m+1} \tag{16}$$

$$\dot{\boldsymbol{p}}_1(0) = \boldsymbol{0}, \quad \dot{\boldsymbol{p}}_M(T_M) = \boldsymbol{0} \tag{17}$$

$$\dot{\boldsymbol{p}}_m(T_m) = \dot{\boldsymbol{p}}_{m+1}(0), \quad \ddot{\boldsymbol{p}}_m(T_m) = \ddot{\boldsymbol{p}}_{m+1}(0) \tag{18}$$

$$T_m \geq 0, \tag{19}$$

where $\mu > 0$ is time penalty factor. The time penalty factor influences the trade-off between minimizing snap and total trajectory time.

4 EXPERIMENTAL SETUP

The quadrotor platform used is the Parrot Bebop 1, and is equipped with a Parrot P7 dual-core Cortex A9 CPU and an MPU6050 IMU. Relevant parameters of the Parrot Bebop 1 are included in Table 1.

The Parrot Bebop 1 has a thrust-to-weight ratio of $T/W \approx 1.7$ significantly lower than that of modern custom drones, limiting the maximum speeds that can be reached. Despite this drawback the Parrot Bebop 1 has certain advantages for our application: (I) the aging onboard CPU has limited computational capacity forcing the online implementation of the adaptive neural controller to be computationally efficient (II) the relatively large body fairing introduces complex aerodynamic disturbances even at low speeds which are hard to model (III) it is equipped out-of-the-box with rpm sensors on all propellers.

Table 1: Parrot Bebop 1 Parameters

Parameter	Unit	Value
m	$[kg]$	0.390
(l_x, l_y)	$[m]$	$(9.5e^{-2}, 7.75e^{-2})$
$\boldsymbol{I}$	$[gm^2]$	$diag\left(9.06e^{-4}, 1.24e^{-3}, 2.05e^{-3}\right)$
c_t	$[N^2]$	$4.36e^{-8}$
c_τ	$[Nm^2]$	$6.60e^{-12}$
(k_x, k_y, k_z)	$[kg/s]$	$\left(1.08e^{-5}, 9.65e^{-6}, 2.79e^{-5}\right)$
k_h	$[kg/m]$	$6.26e^{-2}$

The onboard software of the Parrot Bebop 1 has been flashed with the Paparazzi-UAV open source autopilot project[1] written in C. Designed with autonomous applications in mind and highly customizable, it allows for the straightforward implementation of modules onboard. The offline DNN training is performed in Python using Pytorch[2] an open-source deep learning framework in combination with WandB[3] a powerful tool for managing and tracking machine learning experiments.

[1] https://wiki.paparazziuav.org/wiki/Main_Page
[2] https://pytorch.org/
[3] https://wandb.ai/site

http://www.imavs.org/

The test flights are performed in the CyberZoo, an indoor $10 \times 10 \times 7\ m^3$ square volume equipped with an OptiTrack motion capture system. The OptiTrack measurements are fused with the IMU measurements onboard using an Extended Kalman filter (EKF) for accurate state estimation. A diagram of the experimental setup is shown in Figure 3.

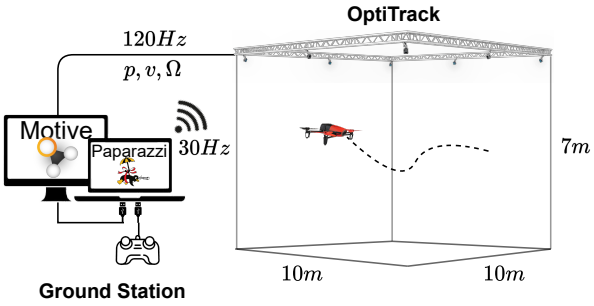


Figure 3: Experimental Setup

5 RESULTS

5.1 Dataset

5.1.1 Data Collection

The DAIML algorithm requires data to train the representation ϕ. Data collection consists of logging relevant time-stamped drone states as the quadrotor tracks minimum snap trajectories using a conventional velocity PID controller. These trajectories are generated from 10 randomly sampled waypoints using the method outlined in subsection 3.3. If at any point throughout the generated trajectory, the position and a tracking margin exceed the limits of the CyberZoo a different set of waypoints is sampled till a suitable trajectory is found. To gather data across a wide range of velocities several minimum snap trajectories are generated with varying time factors. However, all trajectories will travel roughly the same distance but with varying times, resulting in slower trajectories having more data points than faster trajectories. To ensure a more uniform distribution of data points across velocities multiple trajectories are logged at higher speeds, preventing the network from learning a bias towards a certain velocity regime due to it being over-represented in the data collected.

To simulate propeller damage all 4 propellers of the bebop are cut at the tip by approximately $2mm$ and $4mm$ corresponding to a 5.3% and 10.5% reduction in blade radius. Replacing the smooth elliptical propeller tips with rough rectangular tips increases the wing tip vortices generated by the fast-spinning propellers. This results in a reduction in propeller lift and an increase in propeller drag, altering the aerodynamic effects acting on the quadrotor. Data is collected on the training and validation trajectories outlined in Table 2 for all 3 propeller conditions: (1) no damage, (2) slight damage ($2mm$) and (3) significant damage ($4mm$).

Table 2: Training and Validation Dataset

	Time Factor (μ)	#	Total Time (T_{tot}) [s]	Datapoints
Train	0.2	1	270	7547
	0.4	2	263	7486
	0.6	3	311	8883
	0.8	4	282	7968
	1.0	5	287	7923
	1.2	4	205	5691
Val	0.3	1	182	4815
	0.7	1	81	2120
	1.1	1	49	1223
Total	-	22	1,930	61,624

5.1.2 Data Processing

The unmodeled forces acting on the quadrotor are obtained by rearranging Equation 1 to solve for f_{res}. The quadrotor acceleration is required but not logged, instead, it is computed using 1st-order central difference of the velocity measurements. However, the velocity measurements contain significant noise which is further amplified when calculating the acceleration derivative. A zero-phase "filtfilt" filter is used to smooth out the velocity measurements to reduce the noise in the unmodeled force without introducing lag.

The dataset is divided into the 3 different propeller conditions $\boldsymbol{D} = \{\boldsymbol{D}^1, \boldsymbol{D}^2, \boldsymbol{D}^3\}$ with each dataset containing N_k input-output pairs $\boldsymbol{D}^k = \{\boldsymbol{x}_i^k, \boldsymbol{y}_i^k\}_{i=1}^{N_k}$ of the relevant drone states and a noisy measurement of the unmodeled forces $\boldsymbol{y} = \boldsymbol{f}_{res} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ encompasses all source of noise.

5.2 Training

The DAIML algorithm outlined in subsubsection 3.2.1 is used to train an effective representation $\boldsymbol{\phi}$ of the propeller damage invariant aerodynamic effects from the data collected. The inputs of the network $\boldsymbol{\phi}$ are the velocity, quaternion and rpm states $\boldsymbol{x} = [\boldsymbol{v}, \boldsymbol{q}, \boldsymbol{rpm}]$, while the output representation is concatenated $\boldsymbol{y} = [y_1, y_2, y_3, 1]$ to provide a constant term for the adaptation. The architecture of the network $\boldsymbol{\phi}$ consists of $[11, 50, 60, 50, 4]$ fully connected layers with Rectified Linear Unit (ReLU) activation functions as shown in Figure 4. Similarly, the discriminator network h consists of $[4, 128, 3]$ fully connected layers with ReLU activation functions.

The hyperparameters used during training are reported in Table 3, obtained from the sensitivity analysis. The commonly used Adam optimizer, an extended version of stochastic gradient descent, is used to update the weights of both networks $\boldsymbol{\phi}$ and $\boldsymbol{h}$ with learning rates $lr_\phi = 2e^{-3}$ and $lr_h = 4e^{-4}$ and batch sizes $B^t = 128$ and $B^a = 512$ respectively. The normalization term $\gamma = 17$ ensures that the learned adaptation coefficients remain bounded, important for fast adaptation in the online phase. The regularization term α

http://www.imavs.org/

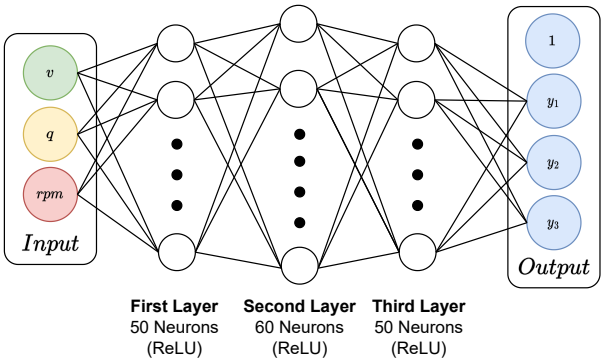


Figure 4: Neural Network ϕ Architecture

influences the degree of adversarial training. A regularization value of $\alpha = 0.1$ was found to balance the ϕ network performance and disentangle the learned linear coefficients into separate propeller conditions. The stability of the discriminator network $\boldsymbol{h}$ is improved by using a discriminator training frequency of $\eta = 0.75$, combined with spectral normalization of the discriminator weights.

Table 3: Hyperparameters of DAIML Algorithm

Hyperparameter	Symbol	Value
ϕ Network		
Learning Rate	lr_ϕ	$2e^{-3}$
Normalization	γ	17
Training Batch Size	B^t	128
$\boldsymbol{h}$ Network		
Learning Rate	lr_h	$4e^{-4}$
Regularization	α	0.1
Training Frequency	η	0.75
Adaptation Batch Size	B^a	512

The high dimensional adaptation coefficients are projected into a 2-dimensional plane using the t-Distributed Stochastic Neighbor Embedding (t-SNE) dimensionality reduction technique. The t-SNE algorithm preserves local similarities in the higher dimensional data enabling visualization of clusters and patterns. The t-SNE plots of the training adaptation coefficients $\boldsymbol{A}^*$ with $\alpha = 0.1$ and $\alpha = 0$ are shown in Figure 7. With $\alpha = 0$ corresponding to the non-adversarial training case. Adversarial training enforces the adaptation coefficients into explaining clusters that contain the propeller damage condition information, ensuring the trained representation ϕ is condition invariant.

5.2.1 Unmodeled Force Prediction

The unmodeled force predictions of the network ϕ are shown on the training, testing and validation data for the no propeller damage condition. The adaptation coefficients learned during training corresponding to the no propeller damage condition are used to compute the unmodeled force predictions $\hat{\boldsymbol{f}}_{ext} = \boldsymbol{\phi}(\boldsymbol{x})\,\boldsymbol{A}^*$.

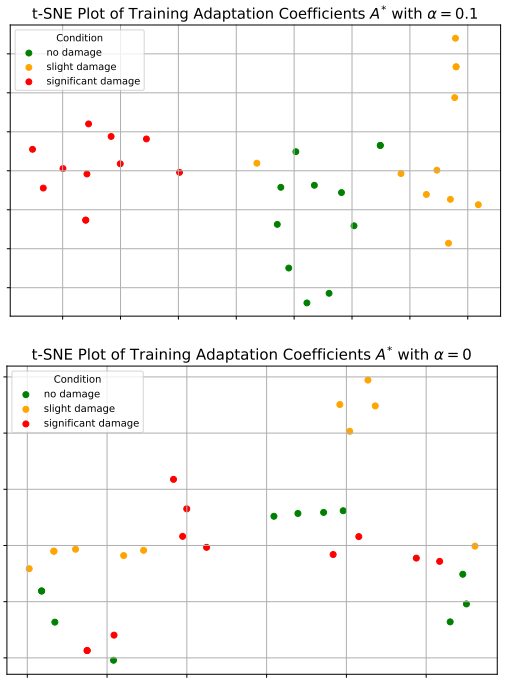


Figure 7: t-SNE plots of the Training Adaptation Coefficients $\boldsymbol{A}^*$ with $\alpha = 0.1$ and $\alpha = 0$

The ϕ network successfully learns a representation that captures the main features of the unmodeled forces acting on the quadrotor, providing a filtering zero-lag estimate of these unmodeled forces. Even at the relatively slow maximum speeds the Parrot Bebop 1 can achieve the conventional aerodynamic model fails to model aerodynamic forces in the order of $f_{res} \approx 0.3$ in the x-y direction and $f_{res} \approx 0.7$ in the z-direction, which the network ϕ can model.

While the network captures the main features of the unmodeled forces they may be off by a certain offset, this is especially noticeable on the validation data. This may be caused by either the different controller architecture between train/test and validation or because a different quadrotor of the same model is used. Substantiating the need to adapt coefficients online to account for differences between training data and real-life deployment.

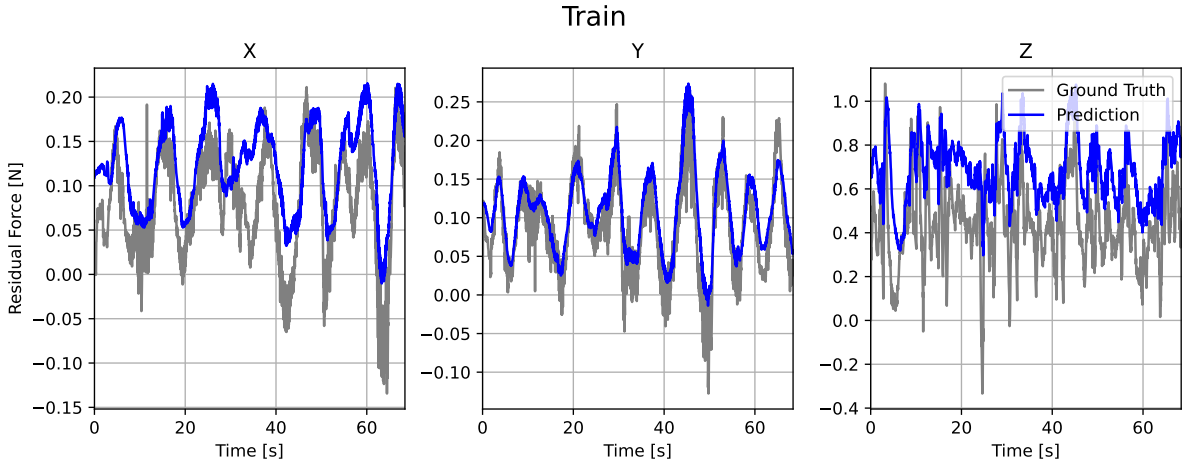
5.3 Tracking Performance

The tracking performance of the neural adaptive controller is compared to that of the nonlinear controller in Equation 8 without the adaptation term and the baseline velocity PID controller. The performance of these controllers is measured by tracking a randomized minimum snap trajectory with time factor $\mu = 0.7$ and $\mu = 1.1$, shown in Figure 9.

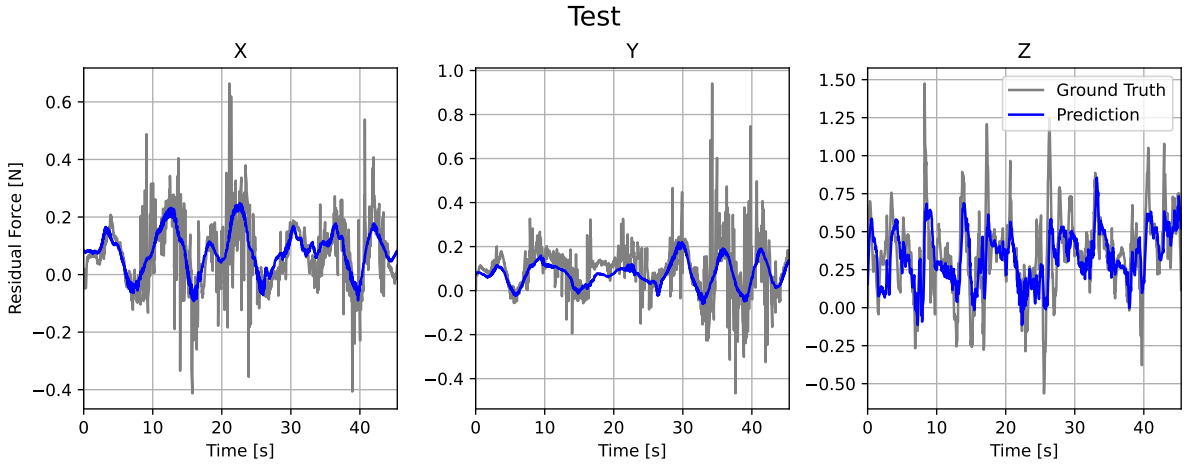
The adaptive neural controller outperforms the other controllers by a substantial margin, achieving a tracking error one order of magnitude lower than the conventional velocity PID controller and approximately $\approx 60\%$ lower than the nonlin-

http://www.imavs.org/

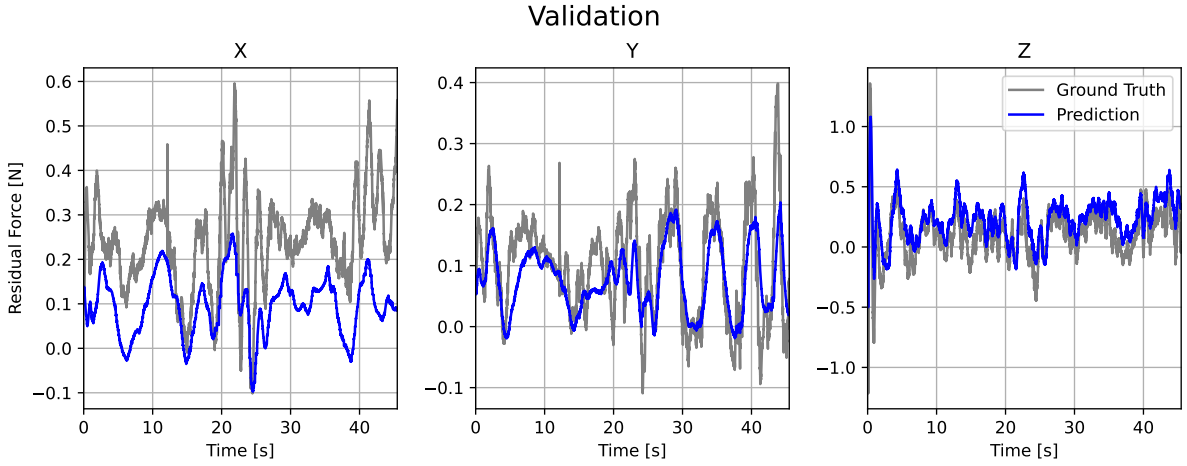
http://www.imavs.org/



(a) Training Data



(b) Testing Data



(c) Validation Data

Figure 8: Unmodeled Force Predictions $\hat{\boldsymbol{f}}_{ext} = \boldsymbol{\phi}(\boldsymbol{x})\,\boldsymbol{A}^{*}$ for No Propeller Damage Condition

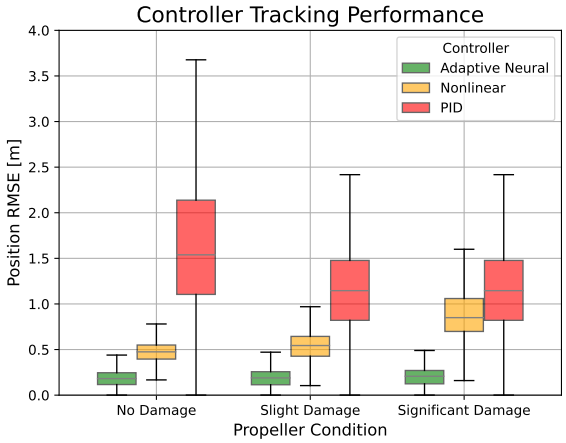


Figure 9: Controller Tracking Performance Comparison on Minimum Snap Trajectory with $\mu = 0.7$

ear controller. Most importantly, the tracking performance of the adaptive neural controller barely degrades with increasing propeller damage in contrast to the other controllers, with a less than 0.03 $[m]$ increase in tracking error. This showcases the ability of the adaptive neural controller to adapt to the current propeller condition flown.

The trajectories flown using the neural adaptive controller and the nonlinear controller are shown in Figure 10 for all propeller damage conditions.

6 Discussion & Recommendations

The adaptive neural controller achieves impressive tracking results with minimal susceptibility toward propeller damage. The neural network ϕ successfully learns a representation of the unmodeled aerodynamic forces acting on the quadrotor.

6.1 *Computational Efficiency*

Most quadrotor adaptive controllers in literature require powerful embedded computers or even a ground station. The adaptive neural controller runs onboard an aging dual-core CPU without breaking a sweat, a testament to its computational efficiency. The efficient C implementation of the network inference and adaptation allows the controller to run onboard at ≈ 30 $[Hz]$ with a CPU load of $\approx 60\%$ but could even run at a much faster ≈ 100 $[Hz]$ at a CPU load of $\approx 80\%$. For the offline learning phase, the DAIML algorithm is equally fast at learning an efficient representation of the aerodynamic forces taking approximately $\approx 4m$ to train a network from scratch on a laptop.

6.2 *Influence of Propeller Damage on Aerodynamic Model*

The effect of propeller damage on the residual forces is mainly limited to the vertical direction, with little change in the lateral residual forces. This causes some instability in the adversarial training procedure albeit these problems could be solved through careful hyperparameter selection. Increasing speed and propeller rpms or increasing propeller damage should make the influence of propeller condition more noticeable, however, we are ultimately limited by the thrust-to-weight ratio of the quadrotor and the available space of the CyberZoo.

6.3 *Noisy Measurements*

The ϕ network is susceptible like any other neural network to the quality of the data it trains on. To compute the unmodeled force measurements acting on the drone the velocity measurements are differentiated, which further amplifies the noise. The OptiTrack velocity measurements experienced high noise, frequency dropouts and occasional asynchronous timing. Despite efforts to filter and clean the training and testing data some of noise and errors made it through into the ground truth data, degrading the quality of the learned representation ϕ.

The poor quality of the OptiTrack measurements also influences the online adaptation. Significant filtering of the velocity measurements is required before applying backward finite difference to obtain usable measurements of the residual force $\boldsymbol{y}$ which have significant lag. This may cause oscillatory behaviour as can be seen in some of the trajectories in Figure 10.

6.4 *Recommendations*

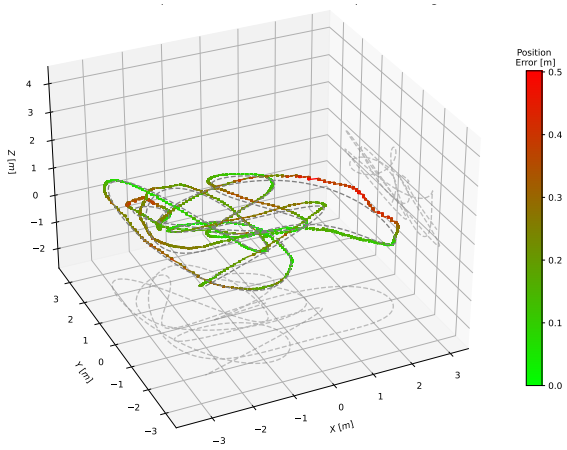
Future work should focus on addressing the challenge of significant noise in the residual force measurements in the online adaptation. The quality of the residual force measurements could be improved by implementing a properly tuned Kalman-filter with accelerations as states. Additionally, accelerometer measurement noise can be mitigated by using a modern IMU or in the case of the OptiTrack velocity measurements using ball infrared markers and limiting reference trajectories to remain in areas with good OptiTrack coverage.

In terms of improving the architecture of the adaptive neural controller, research in artificial intelligence has shifted towards using transformers which outperform conventional fully connected or convolutional network architectures. Transformers process a sequence of inputs in contrast to a single input. Furthermore, transformers have recently been employed in Generative Adversarial Networks in [29] similar to DAIML achieving state-of-the-art performance.

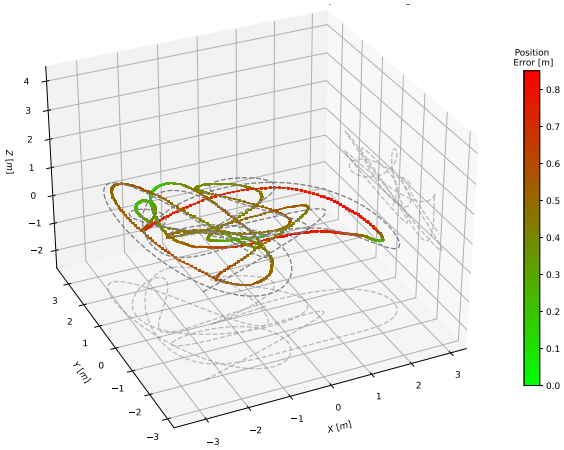
The effect of propeller damage on the residual forces was found to not be as significant as expected, diminishing the benefits of using an adaptive controller. Instead adapting to flying the Parrot Bebop 1 with and without bumpers attached would be a better application. We believe that this adaptive neural controller would be best suited for morphing or VTOL UAVs which have multiple distinct aerodynamic models which are challenging to model during the transition phase.

http://www.imavs.org/

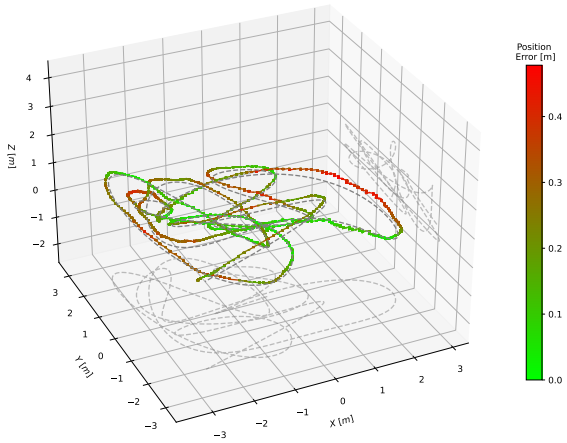
http://www.imavs.org/



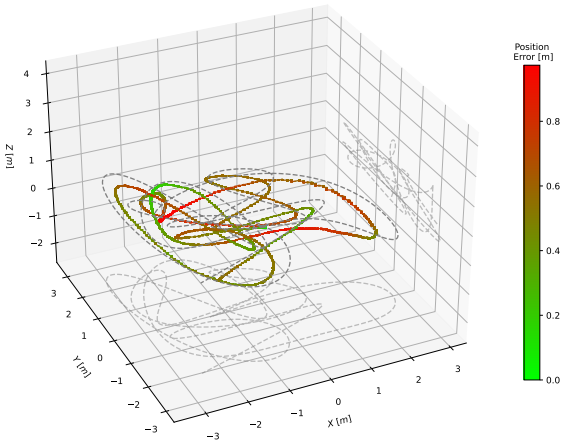
(a) Adaptive Neural Controller w/ No Propeller Damage



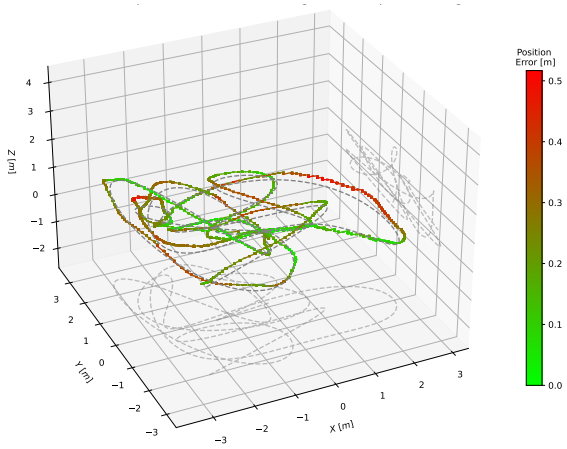
(b) Nonlinear Controller w/ No Propeller Damage



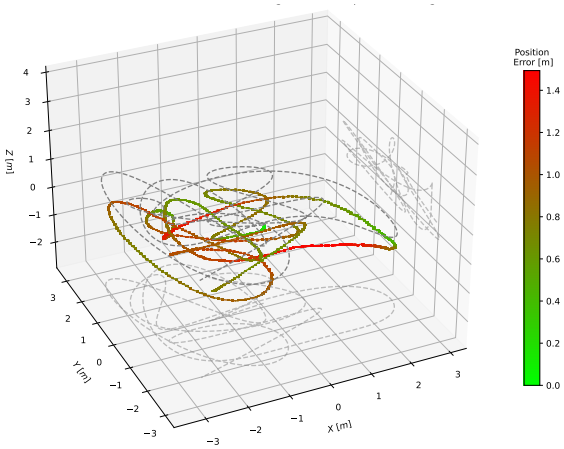
(c) Adaptive Neural Controller w/ Slight Propeller Damage



(d) Nonlinear Controller w/ Slight Propeller Damage



(e) Adaptive Neural Controller w/ Significant Propeller Damage



(f) Nonlinear Controller w/ Significant Propeller Damage

Figure 10: Comparison of Trajectories flown using Adaptive Neural Controller and Nonlinear Controller for all Propeller Conditions

http://www.imavs.org/

7 CONCLUSION

The "Neural-Fly" controller from [27] is a lightweight yet powerful adaptive neural controller that combines pre-trained condition invariant representations of the unmodeled forces using Domain Adversarially Invariant Meta-Learning (DAIML) with fast online adaptation. While it was originally applied to wind speed conditions, we apply this adaptive neural architecture to various degrees of propeller damage a common condition in real-work flight, demonstrating the versatility of this controller architecture in adapting to various conditions. We show that the adaptive neural architecture is a computationally efficient and accurate trajectory-tracking controller architecture. The adaptive neural controller is deployed fully onboard the Parrot Bebop 1 without the need to attach powerful embedded computers, showcasing the computational efficiency of our controller implementation. We perform a sensitivity analysis on the hyperparameters of the DAIML algorithm consisting of 30 runs to determine the optimal hyperparameters using Bayesian search, further improving the performance of the controller. We show that the learned propeller damage invariant representation ϕ using DAIML successfully predicts a filtered zero-lag prediction of the unmodeled forces even at the limited speeds and propeller rpms of the experimental setup. The adaptive neural controller successfully adapts to the various degrees of propeller damage with minimal degradation in tracking performance significantly outperforming the nonlinear and velocity PID baseline controllers.

REFERENCES

[1] Theodore Tzanetos, MiMi Aung, J. Balaram, Havard Fjrer Grip, Jaakko T. Karras, Timothy K. Canham, Gerik Kubiak, Joshua Anderson, Gene Merewether, Michael Starch, Mike Pauken, Stefano Cappucci, Matthew Chase, Matthew Golombek, Olivier Toupet, Marshall C. Smart, Stephen Dawson, Erick Blandon Ramirez, Johnny Lam, Ryan Stern, Nacer Chahat, Joshua Ravich, Robert Hogg, Benjamin Pipenberg, Matthew Keennon, and Kenneth H. Williford. Ingenuity mars helicopter: From technology demonstration to extraterrestrial scout. In *2022 IEEE Aerospace Conference (AERO)*. IEEE, March 2022.

[2] Dara Kerr and Richard Nieva. Drones, sun — and a strong will — elevate Rwanda's health care.

[3] Guillem Torrente, Elia Kaufmann, Philipp Foehn, and Davide Scaramuzza. Data-Driven MPC for Quadrotors. In *IEEE Robotics and Automation Letters*, volume 6 of *2*, pages 3769–3776. IEEE, March 2021.

[4] James Svacha, Kartik Mohta, and Vijay Kumar. Improving quadrotor trajectory tracking by compensating for aerodynamic effects. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 860–866, June 2017.

[5] Patricia Ventura Diaz and Steven Yoon. High-Fidelity Computational Aerodynamics of Multi-Rotor Unmanned Aerial Vehicles. In *2018 AIAA Aerospace Sciences Meeting*, Kissimmee, Florida, January 2018. American Institute of Aeronautics and Astronautics.

[6] Ivan Lopez-Sanchez, Francisco Rossomando, Ricardo Pérez-Alcocer, Carlos Soria, Ricardo Carelli, and Javier Moreno-Valenzuela. Adaptive trajectory tracking control for quadrotors with disturbances by using generalized regression neural networks. *Neurocomputing*, 460:243–255, October 2021.

[7] Mahdis Bisheban and Taeyoung Lee. Geometric Adaptive Control with Neural Networks for a Quadrotor UAV in Wind fields, March 2019.

[8] Qiyang Li, Jingxing Qian, Zining Zhu, Xuchan Bao, Mohamed K. Helwa, and Angela P. Schoellig. Deep Neural Networks for Improved, Impromptu Trajectory Tracking of Quadrotors, October 2016.

[9] Aminurrashid Noordin, Ariffanan Basri, and Zaharuddin Mohamed. Simulation and experimental study on PID control of a quadrotor MAV with perturbation. *Bulletin of Electrical Engineering and Informatics*, 9, October 2020.

[10] Luís Martins, Carlos Cardeira, and Paulo Oliveira. Linear Quadratic Regulator for Trajectory Tracking of a Quadrotor. *IFAC-PapersOnLine*, 52(12):176–181, January 2019.

[11] J.J.E. Slotine and W. Li. *Applied Nonlinear Control*. Prentice Hall, 1991.

[12] Daewon Lee, H. Jin Kim, and Shankar Sastry. Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter. *International Journal of Control, Automation and Systems*, 7(3):419–428, June 2009.

[13] P.V. Kokotovic. The joy of feedback: nonlinear and adaptive. *IEEE Control Systems Magazine*, 12(3):7–17, June 1992. Conference Name: IEEE Control Systems Magazine.

[14] S. Sieberling, Q. P. Chu, and J. A. Mulder. Robust Flight Control Using Incremental Nonlinear Dynamic Inversion and Angular Acceleration Prediction. *Journal of Guidance, Control, and Dynamics*, 33(6):1732–1742, November 2010. Publisher: American Institute of Aeronautics and Astronautics.

[15] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011*

IEEE International Conference on Robotics and Automation, pages 2520–2525, 2011.

[16] Matthias Faessler, Antonio Franchi, and Davide Scaramuzza. Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories. In *Robotics and Automation Letters (RA-L)*, volume 3 of *2*, pages 620–626. IEEE, April 2018.

[17] Ezra Tal and Sertac Karaman. Accurate Tracking of Aggressive Quadrotor Trajectories using Incremental Nonlinear Dynamic Inversion and Differential Flatness. In *IEEE Transactions on Control Systems Technology*, volume 29 of *3*, pages 1203–1218. IEEE, June 2020. arXiv:1809.04048 [cs] type: article.

[18] Sihao Sun, Angel Romero, Philipp Foehn, Elia Kaufmann, and Davide Scaramuzza. A Comparative Study of Nonlinear MPC and Differential-Flatness-Based Control for Quadrotor Agile Flight. In *IEEE Transactions on Robotic*, pages 3357–3373. IEEE, December 2022. arXiv:2109.01365 [cs] type: article.

[19] Prasanth Kotaru, Ryan Edmonson, and Koushil Sreenath. Geometric L1 Adaptive Attitude Control for a Quadrotor Unmanned Aerial Vehicle, March 2020.

[20] Drew Hanover, Philipp Foehn, Sihao Sun, Elia Kaufmann, and Davide Scaramuzza. Performance, Precision, and Payloads: Adaptive Nonlinear MPC for Quadrotors. *IEEE Robotics and Automation Letters*, 7(2):690–697, April 2022.

[21] Zachary T. Dydek, Anuradha M. Annaswamy, and Eugene Lavretsky. Adaptive Control of Quadrotor UAVs: A Design Trade Study With Flight Evaluations. *IEEE Transactions on Control Systems Technology*, 21(4):1400–1406, July 2013. Conference Name: IEEE Transactions on Control Systems Technology.

[22] Xichen Shi, Patrick Spieler, Ellande Tang, Elena-Sorina Lupu, Phillip Tokumaru, and Soon-Jo Chung. Adaptive Nonlinear Control of Fixed-Wing VTOL with Airflow Vector Sensing. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5321–5327, May 2020. ISSN: 2577-087X.

[23] Alexander Spitzer and Nathan Michael. Feedback Linearization for Quadrotors with a Learned Acceleration Error Model. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6042–6048, May 2021.

[24] Fu-Chuang Chen and H.K. Khalil. Adaptive control of a class of nonlinear discrete-time systems using neural networks. *IEEE Transactions on Automatic Control*, 40(5):791–801, May 1995.

[25] A. Kondratiev and Y. Tiumentsev. Inverse Dynamics Approach to Adaptive Damage-Tolerant Control for Unmanned Aerial Vehicles. In *Proceedings of the International Micro Air Vehicle conference and flight competition 2011 summer edition*, 2011.

[26] Suneel Belkhale, Rachel Li, Gregory Kahn, Rowan McAllister, Roberto Calandra, and Sergey Levine. Model-Based Meta-Reinforcement Learning for Flight with Suspended Payloads. *IEEE Robotics and Automation Letters*, 6(2):1471–1478, April 2021.

[27] Michael O'Connell, Guanya Shi, Xichen Shi, Kamyar Azizzadenesheli, Anima Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural-Fly Enables Rapid Learning for Agile Flight in Strong Winds. In *Science Robotics*, volume 7 of *66*, page eabm6597. American Association for the Advancement of Science, May 2022.

[28] Robin Ferede, Guido C. H. E. de Croon, Christophe De Wagter, and Dario Izzo. An adaptive control strategy for neural network based optimal quadcopter controllers, 2023.

[29] Drew A. Hudson and C. Lawrence Zitnick. Generative Adversarial Transformers, March 2022.

http://www.imavs.org/

Binary Networks and Continual Learning for Pose Estimation from a Single Aerial Image

Aldrich A. Cabrera-Ponce[1], L. Oyuki Rojas-Perez[2], Manuel Martin-Ortiz[1], and Jose Martinez-Carranza[2]*

[1] Benemérita Universidad Autónoma de Puebla (BUAP), Puebla, México
[2]Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), Puebla, México

http://www.imavs.org/

ABSTRACT

Pose estimation using aerial images captured by Unmanned Aerial Vehicles (UAVs) allows the localisation in GPS-denied scenarios. Several methods based on deep learning approaches with convolutional neural networks (CNN) have become tools for estimating localisation from images. However, building a model that can estimate the pose from a single image needs a large dataset and training time to obtain a result. Besides, the model can be inappropriate in assessing the correct pose in dynamic scenarios with multiple changes. Therefore, we propose a methodology using a binary network with a Continual Learning (CL) strategy to create an estimation model during the same flight mission. Also, we use a submap scheme and multiple models to acquire the UAV's localisation into different parts of the trajectory. Finally, we use PoseNet, ORB-SLAM2 and single-model for comparison purposes in four scenarios, achieving a percentage error of 14% of the total trajectory and a processing time of 51 ms with our proposed approach.

1 INTRODUCTION

Pose estimation has been implemented in diverse applications within robotics using LiDAR sensors, IMU, and monocular cameras. The latter has gained significant relevance due to the use of images captured by UAVs, providing better information about the environment during a flight mission. With these images, deep learning (DL) methods using convolutional neural networks (CNN) have leveraged visual information to create pose estimation models and thus enable localisation from images. Nowadays, these methods have been employed in UAV tasks such as inspection [1], crop area monitoring [2], visual mapping [3], among others.

PoseNet [4] was the first CNN designed for camera localisation using a single image. Its success inspired the community to develop CNN architectures to improve training time and accuracy, especially in domains such as geo-localisation applications with UAVs [5]. However, these architectures typically demand extensive datasets and training time to establish a reliable model. Consequently, new strategies have been explored to generate estimation models during the same flight mission. Continual learning has emerged as an efficient method for creating these models, requiring less time and data.

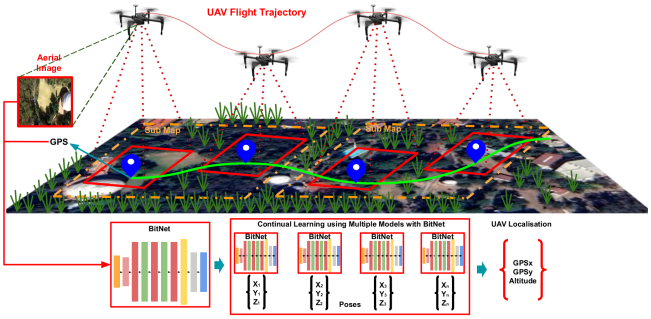


Figure 1: BitNet for aerial localisation: We propose using a binary network to accelerate the training process. In addition, we implement continual learning based on progressive training to extend the knowledge into a multiple-models approach and estimate the pose from a single aerial image.

The continual learning strategy known as latent replay is well-suited for training and updating a model with new information without forgetting prior knowledge. This strategy has proven effective in real-world robotics applications, allowing continuous training while accumulating additional information. However, while continual learning provides efficient pose localisation and time efficiency, accelerating and improving camera pose estimation can be achieved using binary networks. These networks reduce the number of processes, parameters, and calculations in training through weight and activation binarisation. For instance, in [6,7], authors reduced the size of language model weights to one-third of their original size by transitioning from 32-bit to 8-bit representations.

In this work, we propose a methodology that utilises continual learning with a binary network to expedite the training process for obtaining an estimation model. We aim to achieve UAV localisation as a contingency measure in scenarios where GPS signals fail, ensuring uninterrupted flight

*Email address: carranza@inaoep.mx

missions. Based on prior research and the SLAM systems' sub-mapping approach, we support using multiple models along the UAV's flight trajectory. This strategy leverages data acquisition to develop localised models for each flight path segment. Our methodology provides new insights into aerial localisation, by integrating a binary network with continual learning for pose estimation during flight missions. Figure 1 illustrates our aerial localisation methodology using BitNet and continual learning.

This paper is organised as follows. Section 2 shows related works based on continual learning methods; Section 3 outlines the methodology employed, dataset generation and binarised continual training; Section 4 discusses experiments and results obtained. Finally, conclusions and future work are mentioned in Section 5.

2 RELATED WORK

Aerial localisation has been challenging in scenarios with recurrent changes, especially for UAVs conducting flight missions in previously visited environments. Therefore, various methods have been developed to acquire position from an aerial image and use it to locate the UAV when the GPS signal fails. CNN architectures such as PoseNet [4] have enabled significant advancements in camera pose estimation in metres using only one monocular camera, which can be applied to the development of geo-localisation systems [5]. Some compact architectures of PoseNet have been developed to acquire position quickly in outdoor scenarios, such as CompactPN [8], and in indoor environments with DeepPilot4Pose [9].

However, these networks require a lengthy process and an extensive dataset to obtain an estimation model. Therefore, techniques of continual learning have been explored to accelerate the learning process and obtain a model while continuing to collect information. One of the most used strategies is latent replay [10], which involves storing previous information to combine later with new information in external memory. This method avoids catastrophic forgetting and is employed in real-time robotic tasks. In [11], an analysis of continual learning techniques is presented, focusing on human teaching styles. It concludes that teaching experiences do not significantly affect the learning style performed by a robot. Another application of continual learning is presented in [12], where a traversability estimation model adapts a robot to new environments using experience replay with uncertainty, continuously updating the model with new data.

In robotics, systems like Visual Odometry (VO) and SLAM (Simultaneous Localisation and Mapping) have been employed alongside continual learning. Visual Odometry, for instance, is a localisation technique based on camera movement, utilised in continual learning to enable UAV localisation through LiDAR systems in GPS-denied environments [13]. When combined with experience replay, VO facilitates the creation of efficient models using images and dual networks to achieve localisation in diverse scenarios [14]. However, these methods still face the challenge of catastrophic forgetting, as discussed in [15], which examines VO-based continual learning models. This study identifies information loss with minor scene changes, endorsing latent replay strategies that better preserve information without forgetting previous knowledge.

On the other hand, unlike VO, SLAM systems, such as ORB-SLAM2 [16], create a map of the environment, which has been utilised to expand the map of scenarios and achieve UAV localisation in unknown environments [17]. Another approach, Continual SLAM [18], leverages point clouds, images, and loop closure to predict new trajectories using dual networks. Additionally, in [19], researchers utilise dual networks as feedback with linear optimisers to improve the camera pose's visual prediction by identifying BIAS correspondences. However, implementing dual networks and latent replays can incur significant time consumption during the creation and update of estimation models as new information is acquired.

Continual learning finds diverse applications, particularly in localisation systems. In [20], recent research explores continual learning approaches for camera localisation using image inputs. The study emphasises the retention of previous information, the integration of new experiences, and various strategies tailored for localisation, including the use of dictionaries [21]. Furthermore, the research delves into aerial localisation with UAVs, showcasing advancements such as topological localisation based on mean poses [22] and hierarchical search techniques employing sub-maps and multiple models [23]. These studies demonstrate effective UAV localisation using single images, employed for classification-based localisation and recovering poses close to ground truth.

Motivated by previous works, we propose leveraging continual learning with multiple models and submap search combined with BitNet, a binary network initially designed to enhance efficiency and reduce computational costs in Large Language Model (LLM) [6, 7]. This approach opens new possibilities for extending BitNet's capabilities to UAV localisation. By binarising convolutional layers and integrating continual learning, BitNet aims to enrich knowledge and maintain precise position information across diverse scenarios, even in GPS-denied environments.

3 METHODOLOGY

This work aims to develop a localisation system for UAVs using continual learning and a binary network. The methodology involves progressively learning incoming information to expand knowledge through multiple models alongside trajectory. In this way, we generate the dataset and sub-maps for the training process using continual learning strategies. Thus, our methodology aims to estimate aerial poses to localise a UAV within dynamic or GPS-denied scenarios.

http://www.imavs.org/

http://www.imavs.org/

3.1 Dataset Generation

The dataset used for this work is the same presented in [22], which consists of 4 trajectories with fewer than 300 images for the first three of these. In contrast, the fourth trajectory contains 7,000 images with continuous coordinates different from the first ones, whose coordinates are not constant. For this dataset, we used the UAV Matrice 100 connected with the Robot Operating System (ROS) to establish communication between the ground control station (GCS) and the onboard computer. Thus, our configuration enables real-time acquisition while the UAV performs a mission flight.

The training scenarios consist of transversal lengths of 1.0 km for Trajectory 1, 3.0 km for Trajectory 2, 4.6 km for Trajectory 3 and 6.0 km for Trajectory 4. We captured the aerial images with an HD resolution, GPS coordinates, and 50 to 100 metres altitudes. For training purposes, we resized the images to 224×224 and converted the GPS coordinates to metres to handle the flight coordinates better. Besides, we create sub-maps along the trajectory, dividing it into ten sections where each one has 20 to 30 images with coordinates in x, y, and z.

We decided to create sub-maps along the trajectory to train a model for each one, allowing us to manage the localisation in case the UAV loses its position. Thus, we create a sub-map when the UAV moves between 50 to 100 metres from its preceding coordinate, achieving between 5 to 20 flight coordinates for each sub-map. In addition, we generate three keyframes representing the sub-map using indexes, which will be used in the training to identify the corresponding one using the current UAV image. In Figure 2, we show a representation of dataset generation, the creation of sub-maps, the keyframes, and the trajectory of the UAV.

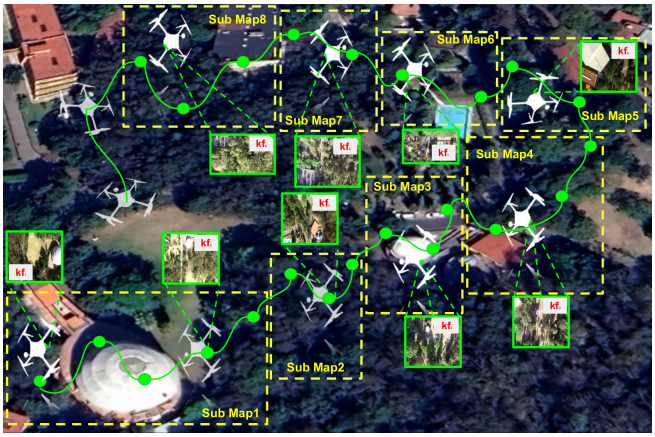


Figure 2: Representation of the UAV trajectory, sub-maps and keyframe generation during a flight mission. We show the UAV trajectory in green, coordinates in green circles, and keyframes in green outlines.

In this way, we generated ten sub-maps for each trajectory, with 30 keyframes for training purposes. In contrast, we generated five keyframes for each sub-map in trajectory four because it is the most extended trajectory, increasing the keyframes for better representation. In this way, we have two datasets, one with aerial images and flight coordinates to train the binary network and the second with keyframes to train a classification network. In Table 1, we show the information about the datasets and the sub-maps generated with the number of images for training.

Table 1: Datasets generated: aerial images include flight coordinates, and keyframes include sub-map indexes.

Traj.	Aerial Images		Keyframes	
	Train	Test	1 Sub-Map	10 Sub-Maps
1	220	144	3	30
2	188	117	3	30
3	236	84	3	30
4	7826	607	5	50

3.2 Binary Network

BitNet is a neural network architecture designed to improve the efficiency and scalability of deep learning models, particularly in applications like large language models [6, 7]. It uses 1-bit representations instead of higher precision formats such as 16-bit or 32-bit, significantly reducing memory requirements and computational load. BitNet employs advanced quantisation techniques to lower the precision of model weights and activations, optimising performance without compromising accuracy. Additionally, it integrates dequantisation techniques to recover information lost during quantisation, preserving as much detail as possible. In some cases, BitNet also employs linear layers instead of convolutional layers to make models lighter and more efficient regarding computational resources.

In this work, we follow the methodology proposed in [6] to implement a binary network, where the binarisation of weights $W \in \mathbb{R}^{n \times m}$ is formulated as:

$$\tilde{W} = \text{Sign}(W - \alpha) \tag{1}$$

Each weight W_{ij} is transformed into $+1$ or -1 based on whether it is greater or less than α, where α represents the average of the weights.

$$\text{Sign}(W_{ij}) = \begin{cases} +1, & \text{if } W_{ij} > 0, \\ -1, & \text{if } W_{ij} \leq 0 \end{cases} \tag{2}$$

$$\alpha = \frac{1}{nm} \sum_{i,j} W_{ij} \tag{3}$$

To ensure that the gradient during backpropagation remains consistent with the original function, we implemented the Straight-Through Estimator (STE) method for $\tilde{W}$. This allows the gradients to be calculated as if no binarisation had occurred, making the training more stable and effective.

The binarisation of activations is described as follows:

$$\tilde{x} = \text{Quant}(x) = \text{Clip}(x \times \frac{Q_b}{\gamma}, -Q_b + \epsilon, Q_b - \epsilon) \quad (4)$$

$$\text{Clip}(x, a, b) = \max(a, \min(b, x)), \quad \gamma = \|x\|_\infty \quad (5)$$

where $Q_b = 2^{b-1}$ for a quantisation of b bits (in our case 8 bits), γ is the maximum absolute value of x, Clip ensures that x is in the range [a, b], and ϵ is a small floating-point number (set to 1×10^{-5}) that prevents overflow during clipping. With the above quantisation equations, the matrix multiplication can be written as:

$$y = \tilde{W}\tilde{x} \quad (6)$$

3.3 Continual Learning

We carried out two continual learning processes for model training using a rehearsal and architectural strategy with aerial images, flight coordinates and sub-maps indexes. The first training was based on an architectural strategy called progressive learning, which focuses on expanding the knowledge within a network architecture. The second training is based on a rehearsal strategy called latent replay, which consists of saving the previous information and combining it with the new one using external memory. Therefore, we use both processes to train dual networks simultaneously during the UAV flight mission.

The first training was a progressive learning process, and instead of expanding the knowledge within a network architecture, we decided to extend it to multiple models. Thus, we trained each model with aerial images and flight coordinates as labels, maintaining the information without suffering catastrophic forgetting. The network is a 4-layer architecture with convolutional layers binarised and a fully connected layer with a regression with three outputs for x, y, and z coordinates. This learning allows us to have a suitable model in case GPS is lost, with the last model responsible for localising the UAV.

Furthermore, this learning strategy allows us to generate a model and acquire new data to create the next one while the UAV perform the flight. Each model was trained with less than 30 images in each sub-map, with 5 to 20 coordinates depending on the scenario. In addition, we perform traditional training using all the images in the dataset to train the binary network, generating a single model of the entire trajectory for evaluation purposes. We train the binary network using 100 epochs, Adam optimiser, and a learning rate of 0.001 for multiple and single models.

We used the keyframes generated in the dataset generation step for the second training. Thus, we use a classification network called InceptioV4 to determine the index and identify the corresponding sub-map using the current aerial image. Therefore, we train the network using the rehearsal strategy based on the latent replay method, which consists of using external memory to keep the previous information. In this way, InceptionV4 extracts maximum and minimum features from the keyframes and stores them in the sample vector whose dimension is ten indexes, each representing a sub-map.

Subsequently, we use this vector to compare the stored features with the features extracted from the current aerial image, thus obtaining the index of the corresponding sub-map. In this way, InceptionV4 allows us to identify the place corresponding to the image to load the binary network model and estimate the pose. The progressive learning and latent replay methods were implemented using a computer laptop with CUDA 12.2, PyTorch 2.3, 8GB of RAM and a GeForce 960M Nvidia card. Thus, our methodology estimates the pose from aerial images, identifying the corresponding sub-map and obtaining the localisation of the UAV. Finally, we show a representative diagram of the continual learning process using aerial images and keyframes in Figure 3.

4 EXPERIMENTS AND RESULTS

We conducted experiments to evaluate the pose estimation and obtain the aerial localisation from a single aerial image. These experiments assess a two-stage process. The first stage consists of searching for the submap corresponding to the current view of the UAV. The second stage aims to estimate the aerial pose using multiple models and establish the UAV localisation into a scenario. For comparison, we evaluate our methodology across four trajectories to estimate the pose frame-by-frame, comparing the error and processing time results with a neural network, PoseNet, and ORB-SLAM2.

4.1 Submap Search Stage

The main idea for the UAV localisation is to find the correct area and estimate the pose using the current aerial image. For that, we present the sub-map search using the features extracted with InceptionV4 and features from a colour histogram. In this way, we find the corresponding submap using the index obtained by matching features between those stored and those extracted from the current image. Thus, if the image matches any keyframe representative of the sub-map, it will return the corresponding index, locating the UAV in a section of the total trajectory.

To show the sub-maps search results, we present a comparison using histogram colour and InceptionV4. In Table 2, we show the number of images in the test dataset used to find the corresponding sub-map, the keyframes found correctly and the accuracy results with each method. We can see that InceptionV4 outperforms the colour histogram due to the better management of the features, with minimums and maximums that grow as the continuous training of the keyframes progresses. Moreover, we argue that using a classification network as a search system can offer advantages due to high-quality features, resulting in the best performance. The results obtained can be used to search the sub-map and load the multiple models of the binary network.

http://www.imavs.org/

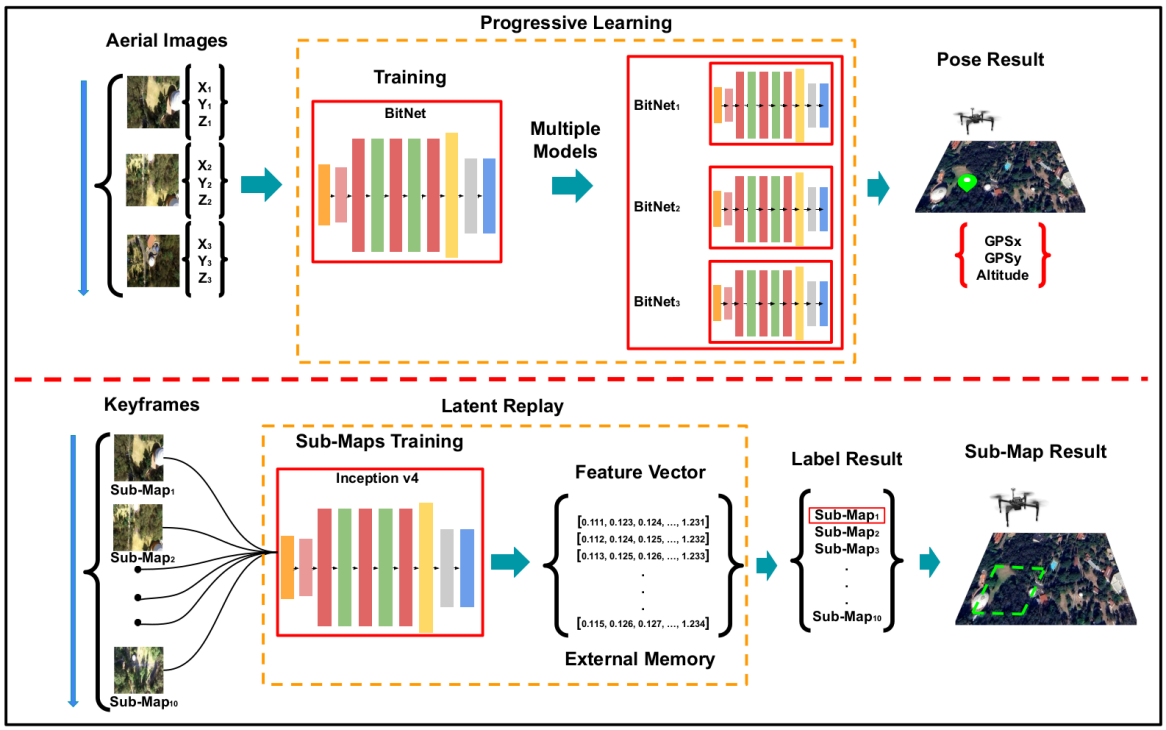


Figure 3: Our training methodology is divided into two parts: Firstly, we train a binary network using aerial images with flight coordinates to create multiple estimation models. Secondly, we train the InceptionV4 network using keyframes to identify the corresponding sub-map.

Table 2: Results of the submap search using colour descriptors and those obtained by the InceptionV4 network.

Traj.	Test	Histogram Colour		InceptionV4	
		Kfs Found	Acc.	Kfs Found	Acc.
1	144	87	0.60	117	0.81
2	117	65	0.55	84	0.71
3	84	56	0.66	60	0.71
4	607	310	0.51	467	0.76

4.2 *Localisation Stage*

This second stage establishes aerial localisation using a single image to estimate the UAV pose. Thus, our methodology works from the moment the UAV images arrive at our localisation system, passing through the InceptionV4 network to find the sub-map index to which it belongs. Once the sub-map is found, the model corresponding to that sub-map is loaded to estimate the pose evaluating the same image. In this way, this process is repeated continuously as each image from the test set arrives. In Figure 4, we illustrate the experimental setup for aerial localisation, showing the sub-maps search, the models' loading and the poses estimation using the input image.

For comparison, we have implemented a 4-layer neural network (NN) to train a single model and multi-model using our methodology. We also used PoseNet and ORB-SLAM2 methods to localise the UAV using a single image. Likewise, we train our binary network using all the images in the dataset to generate a single model and compare it with our multi-models approach. In this way, in Table 3, we present the results obtained with these approaches in the four trajectories, obtaining the Mean Euclidean Distance Error in metres.

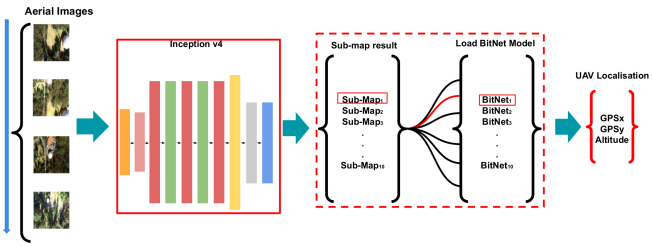


Figure 4: Aerial localisation diagram: The images pass through the InceptionV4 network to identify the corresponding sub-map. Subsequently, we loaded the corresponding estimation model to get the aerial pose.

Similarly, we show the percentage error per axis in Table 4, where a higher percentage consists of a larger error in the pose. These results indicate whether the image obtains the

http://www.imavs.org/

Table 3: Mean Error Euclidean Distances in metres. The best results are highlighted in bold.

Approach	Trajectory 1			Trajectory 2			Trajectory 3			Trajectory 4		
	x	y	z	x	y	z	x	y	z	x	y	z
PoseNet	47.4	15.3	5.07	94.1	82.3	12.7	59.9	38.3	19.0	146.1	130.8	**2.09**
ORB-SLAM2	-	-	-	**13.04**	**10.68**	6.90	-	-	-	-	-	-
NN-Single	39.4	9.72	**0.10**	82.09	75.7	**5.80**	34.4	37.9	**6.06**	140.7	120.6	7.30
NN-Multiple	12.8	10.9	4.07	45.8	48.47	13.5	27.8	**21.5**	7.05	**52.64**	**52.75**	23.2
BitNet-Single	12.3	11.6	3.40	77.2	61.2	12.5	35.7	29.8	8.58	59.1	64.6	17.7
BitNet-Multiple	**9.63**	**5.38**	2.97	33.8	28.8	7.32	**26.3**	21.9	7.39	57.3	57.3	11.9

Table 4: Percentage Error per Axis (%). The best results are highlighted in bold.

Approach	Trajectory 1			Trajectory 2			Trajectory 3			Trajectory 4		
	x	y	z	x	y	z	x	y	z	x	y	z
PoseNet	48.7	30.8	9.91	60.4	43.0	12.7	19.0	33.6	18.9	76.9	53.2	**2.11**
ORB-SLAM2	-	-	-	**16.6**	**5.74**	7.56	-	-	-	-	-	-
NN-Single	40.5	19.5	**5.54**	52.6	39.6	**5.70**	10.9	33.3	**6.03**	74.0	49.0	7.26
NN-Multiple	13.1	22.0	8.07	29.4	25.3	13.4	8.82	**18.8**	7.02	**27.7**	**21.4**	12.1
BitNet-Single	12.7	23.31	6.74	49.58	32.0	12.5	11.3	26.1	8.54	31.1	26.2	17.6
BitNet-Multiple	**9.90**	**10.8**	5.89	21.7	15.0	7.29	**8.34**	19.2	7.35	30.1	23.3	11.9

http://www.imavs.org/

correct aerial localisation, depending on the sub-map identification. Besides, we noted that the z-axis shows the lowest values because all flights are conducted at a consistent height without changes. In contrast, we see a broader percentage of error in the x and y axes due to the low number of images in the dataset, the fast training, and the binary network, which reduces the precision of the training process.

Likewise, we present the total percentage error using each approach in Table 5, which is the sum of the translation error across the entire length of the real trajectory. Thus, we noticed that the ORB-SLAM2 did not obtain the localisation because the aerial images are not continuous, causing the mapping to break. Nevertheless, it achieves the lowest error in trajectory two but does not complete the others, for which we do not obtain a result. In contrast, PoseNet obtained the highest error percentage in all the trajectories due to the low number of images used during the training process. Finally, our multi-model approach with a binary network obtained a mean percentage error of 14.9, showing the lack of precision when performing binary calculations in the training process but being suitable to get the localisation.

Table 5: Total Percentage Error (%). The best results are highlighted in bold.

Approach	Traj.1	Traj.2	Traj.3	Traj.4	Mean
PoseNet	34.3	42.2	22.1	52.0	37.6
Orbslam2	-	11.7	-	-	-
NN-Single	26.3	46.5	14.8	50.0	34.4
NN-Multi	14.1	24.0	10.6	21.9	17.6
BitNet-S	13.8	33.7	13.9	25.3	21.6
BitNet-M	9.11	15.6	10.5	24.6	**14.9**

Finally, we present in Figure 5 the visual results of the localisation using PoseNet, a multi-model approach with a 4-layer network, and our multi-model approach using the binary network. For better visualisation, we show the real trajectory in red and the estimated trajectories in other colours, in which PoseNet is represented in column 1, multiple-models with the 4-layer network in column 2, and our multiple-model approach in column 3. PoseNet presents several estimation jumps, resulting in erroneous localisations, while the 4-layer network approach achieves good estimation results but with some jumps. Finally, our approach estimates the poses closer to the test trajectory, showing better performance than the others.

4.3 Discussions

In order to evaluate the effectiveness of binary networks, it is necessary to know the estimated time to give the pose for each image that arrives. Thus, given the binarisation in the convolutional layers, the training parameters of the network are reduced, making the estimation models have less weight. Furthermore, taking it to a continuous learning process, a model's training and acquisition time can occur during the same UAV flight mission. This process can benefit real-time localisation tasks using images captured with UAVs.

Table 6 presents the total time with the approaches used to estimate the pose and acquire the localisation. Therefore, we evaluate the efficiency of our approach, measuring the processing time in milliseconds (ms) and comparing the entire process from searching the submap to loading the corresponding model to obtain the estimated pose from a single image. PoseNet achieves the slowest processing time of the other approaches, while ORB-SLAM2 only finishes trajec-

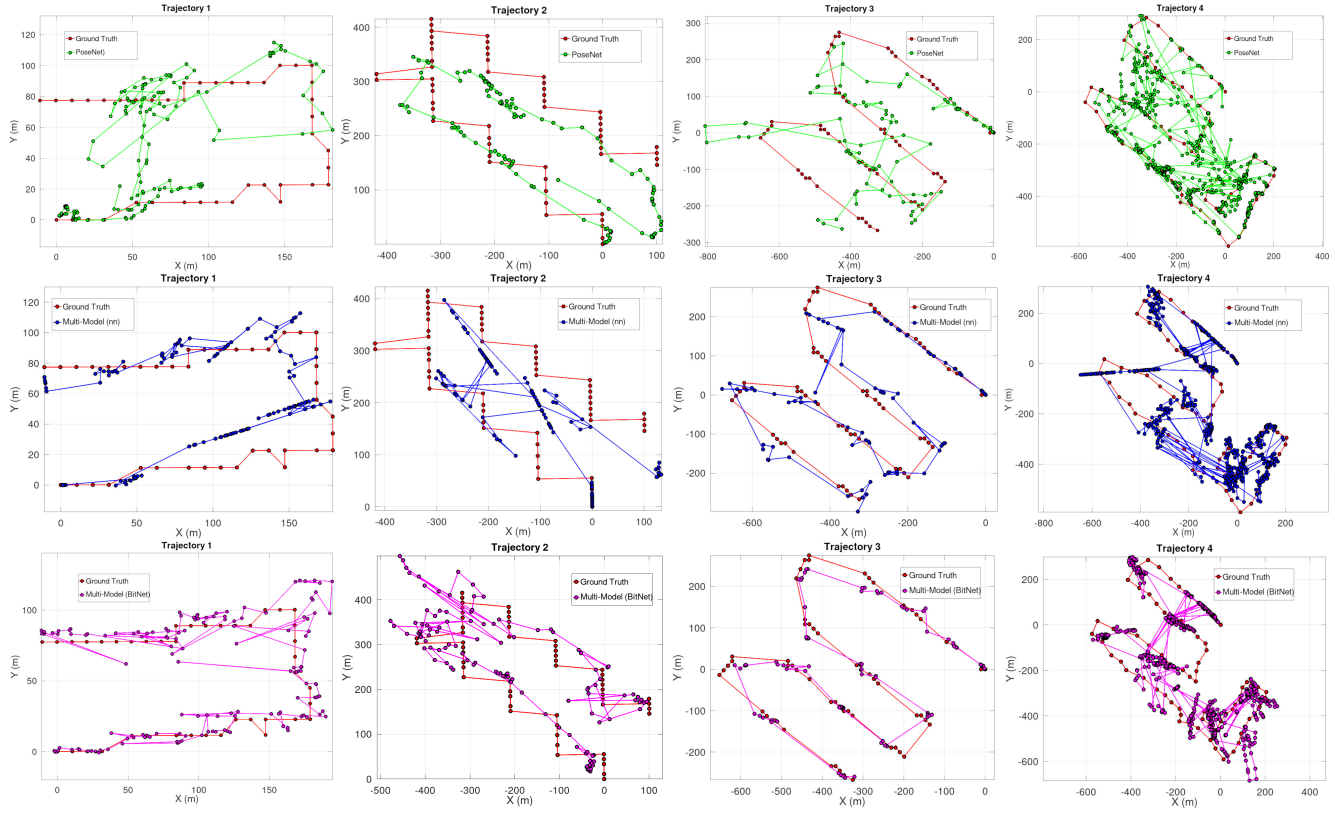


Figure 5: Pose estimation results: we show the testing trajectory in red and circle the ground truth poses. BitNet's multi-model approach obtained better pose estimation results than the other approaches, acquiring the UAV localisation in most images.

tory 2 with a processing time similar to PoseNet. The single model approach with a 4-layer network and BitNet exhibits faster processing time by handling a single model without sub-maps searching to get the aerial pose.

Table 6: Total Processing Time (ms). The best result is highlighted in bold.

Approach	Traj.1	Traj.2	Traj.3	Traj.4	Mean
Orbslam2	-	79.12	-	-	-
PoseNet	71.94	73.74	73.52	72.15	72.83
NN-Single	56.01	50.25	45.22	49.82	50.32
NN-Multi	61.45	61.43	64.60	64.18	62.91
BitNet-S	18.47	19.94	18.73	18.69	**18.96**
BitNet-M	51.80	52.22	49.91	51.85	51.44

Nevertheless, the multi-model approaches with a 4-layer network and BitNet require loading models w.r.t the sub-map index found. Thus, the processing time increases when handling a load of multiple models, even with small 4-layer networks, and the binarisation of the convolutional layers with BitNet. Despite this, our approach achieves good performance and a lower time than PoseNet, which is suitable for real-time localisation tasks with non-continuous and small datasets. Additionally, we can reduce the inference time by handling fewer training bits in the quantisation, but we will take it for future work.

5 CONCLUSION

We proposed a methodology for aerial localisation based on pose estimation with multiple models using BitNet. We leverage the quantisation to binarise the convolutional layers of BitNet, improving the training process to use 8-bit computes instead of float values. Besides, we implemented a localisation by sub-map identification comparing the features of the current images with those of the representative keyframes. This method allows us to load the corresponding BitNet model to estimate the pose from a single image. Our main contribution is the first implementation of BitNet to localisation tasks using aerial images captured by UAV and adapted to a continual training strategy based on progressive learning.

To evaluate the effectiveness of our approach, we compare the poses estimated with PoseNet, ORB-SLAM2, and a 4-layer network. Also, we performed traditional training with all the data to get a single model without sub-map searching. The total percentage error results demonstrated that our approach outperformed the others across all four trajectories,

http://www.imavs.org/

with a percentage error of 14% with multiple models and 21% using a single model. Besides, we achieved a processing time of 51 ms using multiple models and 18 ms with a single model using BitNet.

Finally, implementing a binary network for pose estimation accelerates the processing time using the quantisation method. Furthermore, with progressive learning, we ensure that we retain prior knowledge by using multiple models with flight coordinate information. It should be noted that quantisation reduces the precision of the training, obtaining a suitable result in the UAV localisation. However, the poses help recover the UAV in cases where the GPS signal is lost. In future work, we want to take advantage of and exploit binary networks, improving the processing time and accuracy of pose estimates.

ACKNOWLEDGEMENTS

The authors Aldrich A. Cabrera-Ponce and L. Oyuki Rojas-Perez thank CONAHCYT in Mexico for their scholarships with numbers 802791 and 924254.

REFERENCES

[1] Diego Benjumea, Alfonso Alcántara, Agustin Ramos, Arturo Torres-Gonzalez, Pedro Sánchez-Cuevas, Jesus Capitan, Guillermo Heredia, and Anibal Ollero. Localization system for lightweight unmanned aerial vehicles in inspection tasks. *Sensors*, 21(17):5937, 2021.

[2] David Bohnenkamp, Jan Behmann, and Anne-Katrin Mahlein. In-field detection of yellow rust in wheat on the ground canopy and uav scale. *Remote Sensing*, 11(21):2495, 2019.

[3] Jie Qian, Kaiqi Chen, Qinying Chen, Yanhong Yang, Jianhua Zhang, and Shengyong Chen. Robust visual-lidar simultaneous localization and mapping system for uav. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5, 2021.

[4] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015.

[5] Aldrich A Cabrera-Ponce and J Martinez-Carranza. Aerial geo-localisation for mavs using posenet. In *2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS)*, pages 192–198. IEEE, 2019.

[6] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. Bitnet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453*, 2023.

[7] Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, 2024.

[8] Aldrich A Cabrera-Ponce and Jose Martinez-Carranza. Convolutional neural networks for geo-localisation with a single aerial image. *Journal of Real-Time Image Processing*, 19(3):565–575, 2022.

[9] L Oyuki Rojas-Perez and Jose Martinez-Carranza. Deeppilot4pose: a fast pose localisation for mav indoor flight using the oak-d camera. *Journal of Real-Time Image Processing*, 20(1):8, 2023.

[10] Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent replay for real-time continual learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10203–10209. IEEE, 2020.

[11] Ali Ayub, Zachary De Francesco, Jainish Mehta, Khaled Yaakoub Agha, Patrick Holthaus, Chrystopher L Nehaniv, and Kerstin Dautenhahn. A human-centered view of continual learning: Understanding interactions, teaching patterns, and perceptions of human users towards a continual learning robot in repeated interactions. *ACM Transactions on Human-Robot Interaction*, 2024.

[12] Hyung-Suk Yoon, Ji-Hoon Hwang, Chan Kim, E In Son, Se-Wook Yoo, and Seung-Woo Seo. Adaptive robot traversability estimation based on self-supervised online continual learning in unstructured environments. *IEEE Robotics and Automation Letters*, 2024.

[13] Jiun Fatt Chow, Basaran Bahadir Kocer, John Henawy, Gerald Seet, Zhengguo Li, Wei Yun Yau, and Mahardhika Pratama. Toward underground localization: Lidar inertial odometry enabled aerial robot navigation. *arXiv preprint arXiv:1910.13085*, 2019.

[14] Niclas Vödisch, Daniele Cattaneo, Wolfram Burgard, and Abhinav Valada. Covio: Online continual learning for visual-inertial odometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2464–2473, 2023.

[15] Paolo Cudrano, Xiaoyu Luo, and Matteo Matteucci. The empirical impact of forgetting and transfer in continual visual odometry. *arXiv preprint arXiv:2406.01797*, 2024.

[16] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262, 2017.

http://www.imavs.org/

[17] Ali Safa, Tim Verbelen, Ilja Ocket, André Bourdoux, Hichem Sahli, Francky Catthoor, and Georges Gielen. Learning to slam on the fly in unknown environments: A continual learning approach for drones in visually ambiguous scenes. *arXiv preprint arXiv:2208.12997*, 2022.

[18] Niclas Vödisch, Daniele Cattaneo, Wolfram Burgard, and Abhinav Valada. Continual slam: Beyond lifelong simultaneous localization and mapping through continual learning. In *The International Symposium of Robotics Research*, pages 19–35. Springer, 2022.

[19] Youqi Pan, Wugen Zhou, Yingdian Cao, and Hongbin Zha. Adaptive vio: Deep visual-inertial odometry with online continual learning. *arXiv preprint arXiv:2405.16754*, 2024.

[20] Aldrich A Cabrera-Ponce, Martin-Ortiz Manuel, and Jose Martinez-Carranza. Continual learning for camera localization. *Machine Learning for Complex and Unmanned Systems*, pages 14–33, 2024.

[21] Shuzhe Wang, Zakaria Laskar, Iaroslav Melekhov, Xiaotian Li, and Juho Kannala. Continual learning for image-based camera localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3252–3262, 2021.

[22] Aldrich A Cabrera-Ponce, Manuel Martin-Ortiz, and Jose Martinez-Carranza. Continual learning for topological geo-localisation. *Journal of Intelligent & Fuzzy Systems*, 44(6):10369–10381, 2023.

[23] Aldrich Alfredo Cabrera-Ponce, Manuel Isidro Martin-Ortiz, and Jose Martinez-Carranza. Hierarchical continual learning for single image aerial localisation. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 40–48, Aachen, Germany, Sep 2023. Paper no. IMAV2023-5.

http://www.imavs.org/

Feasibility Study of a Solar Powered Hybrid Micro-Aerial Vehicle

Burhan Saeed *[1] and Peter Thomas [2]
[1] Department of Aeronautics, Imperial College London
[2]School of Physics, Engineering and Computer Science, University of Hertfordshire

http://www.imavs.org/

ABSTRACT

Micro aerial vehicles (MAVs) are finding new applications every day, from surveillance to civil structure health monitoring and exciting photography. Parallel advancements in ultra-thin, high power-to-weight ratio (PWR) solar panels are paving the way for sustained flight, especially for MAVs that are heavily dependent on the limited energy density of batteries. There is renewed interest in solar-powered MAVs, which can offer unprecedented flight times on sunny days. In this context, we propose a solar-powered hybrid MAV configuration, named 'Solar Swifter' that combines the performance of a quadcopter, allowing vertical take-off and landing (VTOL), with the capabilities of a fixed-wing aircraft for conventional cruise flight. Preliminary analysis has shown that with just 1.9 W from ultra-high PWR air-stable solar panels (Organic Photovoltaic, OPV), an MAV with a mass of $51g$ can achieve vertical take-off and fly at $8m/s$. This performance can be further enhanced by using a lower aspect ratio wing, resulting in a square-shaped wing. The next step in developing the Solar Swifter is to investigate the aerodynamic effects of a fan-in-wing configuration at low Reynolds numbers, which has not been studied in the past.

1 INTRODUCTION

Unmanned Aerial Vehicles (UAVs) are rapidly becoming integral to various aspects of everyday life. Among these, Micro Aerial Vehicles (MAVs) have garnered significant attention, particularly for their potential in autonomous applications [1, 2]. A major challenge in achieving full autonomy for MAVs lies in the incomplete understanding of how performance, power, and computational limitations impact their functionality. MAVs must operate within stringent power budgets, severely limiting their flight endurance.
One promising solution to this limitation is the use of solar power. It has been shown that it is feasible to have solar powered micro aerial vehicles [3, 4], albeit, requiring further performance improvements.The performance and application of MAVs are limited by their mode of flight. They are typically either rotary-wing, which allows vertical take-off and landing (VTOL) and hover capability, or fixed-wing, which enables relatively higher payload, range, and endurance capabilities. This paper investigates the performance of a hybrid MAV that combines both rotary and fixed-wing capabilities, aiming to leverage the performance advantages of each. The recent advancements in flexible solar panels [3, 5, 6] which offer high PWRs, provide additional motivation for this approach. By integrating these technologies, the hybrid MAV could potentially enhance both the flight endurance and operational flexibility, making it more suitable for a wide range of applications.

2 AERODYNAMIC CHARACTERISTICS AT LOW REYNOLDS NUMBER

MAVs are classified based on their mass, up to 100 grams, and size, up to 0.1524 m (or six inches). This classification places them in the Reynolds number range between 10^4 and 10^5. This relatively low Reynolds number implies that the aerodynamic efficiency, specifically the lift-to-drag (L/D) ratio, of their wings is much lower compared to larger aircraft, as illustrated in Figure1.
The low L/D ratio at low Reynolds numbers is primarily due to the dominance of viscous forces, early laminar flow separation, inefficient pressure recovery, thicker boundary layers, and reduced lift generation. Understanding these factors is crucial for designing efficient aerofoils and aircraft operating in this regime. The low L/D ratio imposes higher power demands, which is particularly challenging for MAVs, where the power budget is already stringent. Therefore, aerodynamic design requires extra care to mitigate these effects.
The Reynolds number range of 10^4 to 10^5 lies at a transition point where the L/D ratio can vary significantly, potentially changing by an order of magnitude depending on the aerofoil design. Appropriate surface roughness can help maintain linearity in the L/D ratio within this Reynolds number range. However, achieving favourable surface roughness is challenging at the MAV scale.

The Reynolds number has a significant impact on the lift-to-drag (L/D) ratio of a cambered aerofoil, whereas a symmetrical aerofoil experiences relatively minor changes, as shown in Figure 2. For a symmetrical aerofoil, such as the NACA 0003, a five fold increase in Reynolds number (1x10^5)

*Email address(es): b.saeed22@imperial.ac.uk, b.saeed@herts.ac.uk

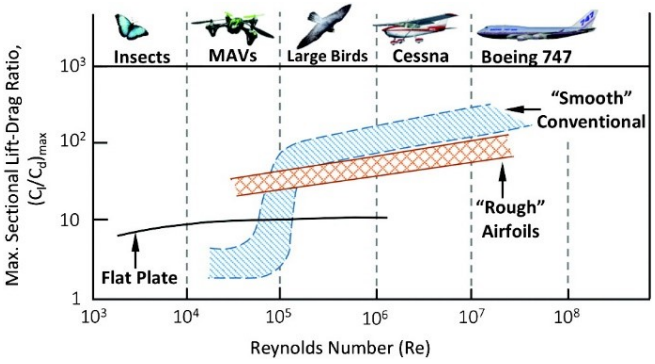


Figure 1: Aerodynamic efficiency of wing-section versus Reynolds number for different regimes of aircraft (taken from [7]).

results in 1.4 times increase in the maximum section lift to drag ratio (C_l/C_d). In contrast, for a cambered aerofoil, like the NACA 6403, the increase in section lift to drag ratio is approximately threefold. Additionally, higher camber aerofoils such as the NACA 6403 offer significantly higher aerodynamic efficiency compared to lower camber aerofoils such as the NACA 4403 and NACA 2403.

3 PERFORMANCE ANALYSIS OF MAVS

Micro Aerial Vehicles (MAVs) primarily utilise three configurations: fixed-wing, rotary-wing, and flapping-wing. Each configuration has distinct characteristics that influence performance and application.

Flapping-wing MAVs are typically very lightweight and offer a few minutes of endurance, making them suitable for specific short-duration tasks [8]. Rotary-wing MAVs, such as the DJI Mavic 3, which weighs 963 grams and can fly for up to 43 minutes [9], are highly maneuverable, providing useful range and endurance. Fixed-wing MAVs, on the other hand, offer the best endurance and range for a given mass. For instance, the Black Widow, weighing 80 grams, can fly for 30 minutes [10], whereas a comparable rotary-wing MAV, such as the QRW100S, weighs 90 grams but only offers 10 minutes of endurance.

Figure 3 summarises the performance of MAVs across these configurations. On average, fixed-wing MAVs provide approximately 2.5 times higher performance compared to rotary-wing MAVs. Flapping-wing MAVs exhibit the best performance for MAVs weighing less than 50 grams, overlapping with the category of Nano Aerial Vehicles (NAVs). This comparison highlights the trade-offs between maneuverability, endurance, and weight among different MAV configurations.

4 SOLAR POWERED AIRCRAFT

The quest for a solar powered aircraft began in 1974, the very first aircraft was named Sunrise which had a span of

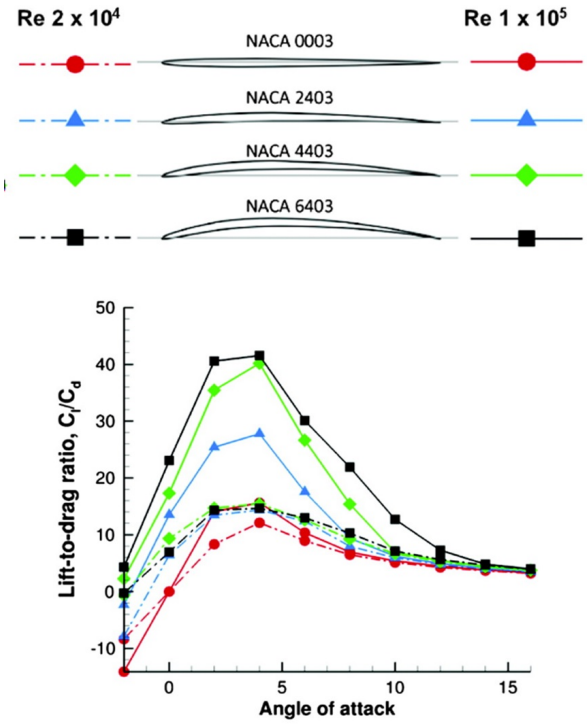


Figure 2: The effect of Reynolds number on different aerofoils used for MAVs (adapted from [7]).

9.75 meters and an empty mass of 12.25 kg [12]. To date over hundred different solar powered aircraft have been built and flown [13], out of those the most prominent aircraft considered to be be milestones of solar flight have been collected in [14] along with their performance data.

Micro Aerial Vehicles (MAVs) have garnered significant attention for their potential in various applications, including surveillance, environmental monitoring, and disaster management. Enhancing their endurance and operational efficiency through the integration of solar panel technology has emerged as a fruitful avenue. Initial research into solar-powered MAVs primarily focused on feasibility studies and conceptual designs. Early efforts were hampered by the limited efficiency of photovoltaic cells and the high weight of available solar panels. Researchers experimented with various MAV designs to optimise the surface area for maximum solar exposure while maintaining aerodynamic efficiency [6].

Silicon-based solar cells, representing the first generation of solar technology, are primarily divided into two types: monocrystalline and polycrystalline silicon. Monocrystalline silicon cells are renowned for their high efficiency, typically ranging from 20% to 25%. However, they are costly to produce and rigid, limiting their applications where flexibility is required. In contrast, polycrystalline silicon cells offer moderate efficiency, between 15% and 20% [15], at a

http://www.imavs.org/

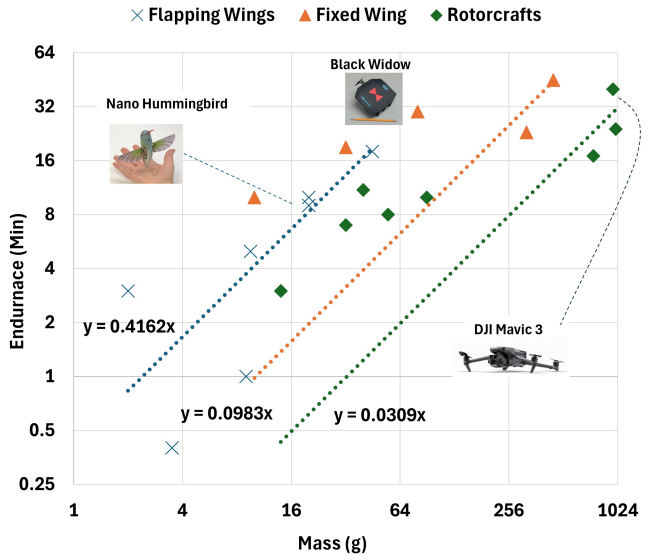


Figure 3: The Endurance performance of MAVs with different configurations (data sources [10, 11, 9]).

lower cost, making them more affordable but less efficient compared to their monocrystalline counterparts.

Thin-film solar cells, considered the second generation of solar technology, include materials such as cadmium telluride (CdTe) and copper indium gallium selenide (CIGS). CdTe cells provide moderate efficiency (10-16%) [16] and perform better in low-light conditions, making them suitable for diverse environmental settings. CIGS cells offer higher efficiency (12-20%) than CdTe and have the added advantage of being flexible, which broadens their application range. However, their production process is more complex. The advent of ultrathin perovskite solar cells marks a significant advancement in solar technology. These cells exhibit rapidly improving efficiencies, with laboratory results exceeding 25%, thus rivaling and potentially surpassing traditional silicon cells. Perovskite solar cells benefit from high absorption coefficients and tunable bandgaps, allowing for tailored applications and the potential for higher efficiencies through tandem structures.

Organic photovoltaics (OPVs) are another emerging technology with current efficiencies lower than silicon and Perovskite cells, typically in the range of 10-15% [17]. Despite this, OPVs offer significant advantages, including their lightweight and flexible nature. They can be manufactured using low-cost, roll-to-roll processes, making them attractive for applications where flexibility and lightweight are crucial, such as in portable and wearable electronics.

Figure 4 highlights 'typical' losses along the route from the sunbeam to the thrust force, including battery charging and discharging, which amounts to a net loss of 85% to 89%. This substantial figure clearly indicates the need for optimising each sub-system to achieve good overall performance.

Theoretical values for maximum efficiency for solar cells range between 15% and 25%, [3] depending on material characteristics, as highlighted above, but these values are not typically achieved in practical production cells.

Most solar cells are mechanically cut silicon slices, and the material is often too thick to be lightweight. To protect these cells from mechanical damage, humidity, and temperature, they must be embedded in a foil or fiberglass. New manufacturing processes have been developed to deposit a thin silicon layer onto a foil, resulting in very lightweight and flexible cells. The efficiency of thin solar cells has considerably improved in recent years, thanks to advancements in material science and fabrication techniques.

Despite these improvements, optimising the entire energy conversion and storage system is crucial to enhancing overall efficiency. This includes improving solar cell efficiency, minimising losses in battery charging and discharging, and enhancing the aerodynamics and weight distribution of MAVs. By addressing these areas, we can significantly reduce energy losses and improve the performance and endurance of solar-powered MAVs.

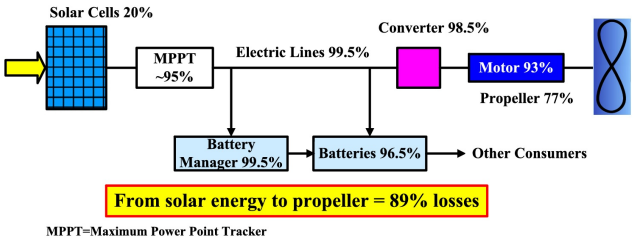


Figure 4: Typical energy losses in a solar powered propulsion system (adapted from [18]).

For the application of solar technologies in MAVs, the most fundamental performance attribute is the power-to-weight ratio (PWR). The PWR of solar panels has evolved rapidly over the past decade. The PWR of different technologies, as illustrated in Figure 5, suggests that OPVs are the highest performers with a PWR as high as 33.8 W/g. The ultrahin Perovskite comes second with a PWR of 30 W/g. However, due to higher solar energy conversion efficiency, the Perovskite solar cells are particularly suitable for applications where weight, efficiency and flexibility are critical factors and that is the MAVs.

5 CONCEPTUAL DESIGN - HYBRID MAV

Having discussed the solar technologies above, it is useful to study successful electric MAVs when designing a solar MAV. Lockheed Martin's MicroSTAR4 has a mass of 85 g, a 23 cm wingspan, and a cruise speed of 11 m/s. The Black Widow, another notable MAV, has a mass of 80 g, a wingspan of 15 cm, and a flight time of 30 minutes. The Black Widow

http://www.imavs.org/

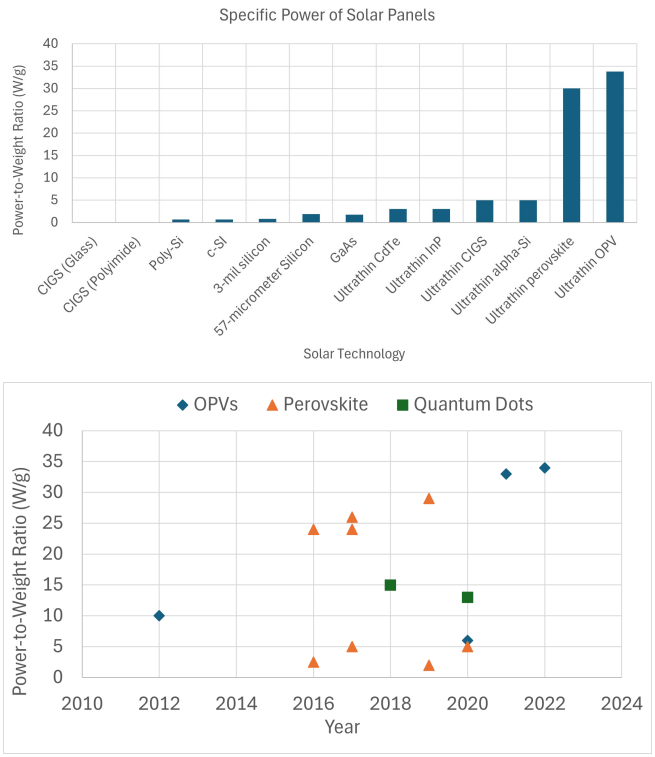


Figure 5: The Power-to-Weight ratio of different solar technologies (data source [7]).

utilises 4.35 W of power from its batteries [10].
It has been shown that propeller efficiencies of 80 percent or greater are achievable at the MAV scale, and motor efficiencies of 70 percent [4, 10] are also possible . Additionally, an aerofoil with a maximum linear dimension of 0.1524 cm can fly at the low Reynolds numbers associated with small aerofoils and low speeds, achieving a lift-to-drag ratio of 10. For the Black Widow, the 80 grams of total mass are divided into approximately 50 grams for propulsion, 10 grams for payload, 7 grams for controls, and 14 grams for the structure. In summary, compiling basic design parameters for solar and electric MAVs is beneficial. 6 lists the wing loading and PWR of different solar-powered aircraft and MAVs. It is clear that the relatively low wing loading, i.e., an ultra-lightweight body, is crucial for the flight capabilities of MAVs.

Solar-powered MAVs are feasible in either rotorcraft or fixed-wing configurations [4, 13, 19]. Combining these two configurations can offer performance benefits, including improved range, endurance, payload capacity, and maneuverability. Figure 7 presents a sketch of the proposed configuration, named 'Solar Swifter SS1'. The main features include solar panels covering the top surface, a propeller attached to the nose of the aircraft providing forward thrust, and triple propellers providing vertical thrust or lift. The triple propellers are attached to the wing tips and the tail, enabling

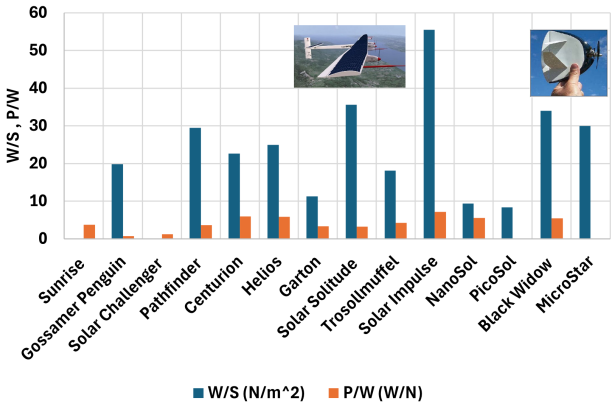


Figure 6: Comparing wing loading and power loading of solar powered aircraft and MAVs (data source [13, 4])

pitch and lateral stability and control by positioning the center of gravity aft of the aerodynamic center. Furthermore, the propellers are ducted or embedded in the wing, which may offer up to 100 percent lift enhancement with a drag penalty [20].

5.1 The Sizing Approach

The MAV class of aircraft lies at the critical Reynolds number regime where lift to drag ratio can jump ten folds which can negatively impact the performance as highlighted above in Figure 1. The sizing can begin by locking the span dimension to 0.1524 m, to comply with the widely endorsed maximum size of MAVs [1, 11, 21]. The relationship of Reynolds number with aspect ratio of wing for different air speeds is plotted in Figure 8, a low aspect ratio, $AR < 2$, for the range of flight speeds considered, $6 < V < 20$ m/s, enables Reynolds number above 1×10^5 to utilise the significant aerodynamic benefits highlighted in the previous section.

In level flight the required thrust depends on the overall drag of the aircraft, the standard drag equation is given below

$$C_D = C_{D_0} + \frac{1}{\pi ARe} {C_L}^2 \tag{1}$$

where C_{D_0} is the zero-lift drag. A conservative estimate of the zero-lift drag coefficient is made, estimating it to be 0.05. This value is derived from considering the friction coefficient for laminar and turbulent flow over a flat plate at a Reynolds number of 1×10^5, which is approximately $0.04 < C_{D_0} < 0.06$ [22]. The lift dependent drag is a function of Oswald efficiency factor e (a factor that represents the change in drag with lift for a three-dimensional wing or aircraft) which is significantly different at the low Reynolds number regime considered herein [23]. At the Reynolds number of 1×10^5 the Oswald efficiency can significantly change with the aspect ratio and the taper ratio λ of the wing, see Figure 9. An AR value of 5 can reduce the Oswald efficiency by

http://www.imavs.org/

http://www.imavs.org/

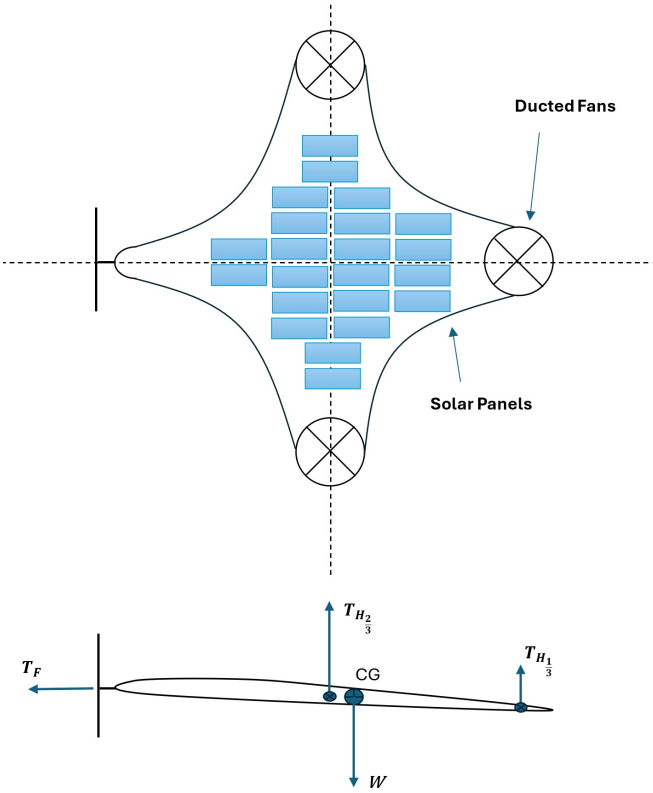


Figure 7: A sketch of the proposed configuration.

50% compared to an AR value of 2. This further consolidates the design point to be around AR of 2.

6 Flight Performance of Solar Swifter SS1

An estimation of Solar Swifter's performance can begin by evaluating the available power, which depends on the efficiency and power loading of the solar panels used. In this context, we employ the recently developed ultra-thin organic photovoltaic (OPV) panels [6], known for their high power-to-weight ratio (PWR) of 33.8 W/g, weight-to-area ratio of 0.0458 N/m$\hat{2}$, and efficiency of 15.8% under a solar irradiance of 1000 W/m$\hat{2}$. Therefore, the available power is directly proportional to the wing's plan area, calculated as $P_{ava} = 0.0458S$ W.

The available power must be sufficient to support both vertical take-off and landing (VTOL) as well as forward flight modes. During vertical flight, the aircraft must overcome gravity and air resistance. Consequently, the required thrust can be determined using the following equation:

$$T_{VTOL} = (W_{pp}+W_{St}+W_{Sys}+W_{Sol})+\frac{1}{2}\rho V_c^2 S C_{D_P} \quad (2)$$

where W_{pp} is the weight of powerplant, W_{St} is the structural weight, W_{Sys} is the systems weight, W_{Sol} is the solar

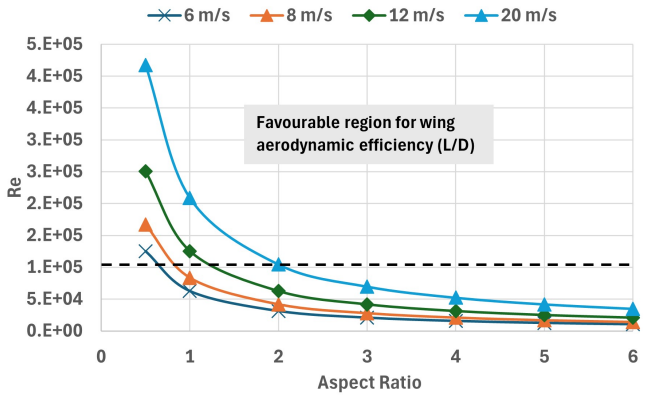


Figure 8: Reynolds number versus aspect ratio of the wing.

panels weight, V_c is the vertical or climb velocity and C_{D_P} is the profile drag coefficient. To evaluate the air drag in vertical flight mode, we assume the aircraft behaves like a flat plate with airflow perpendicular to the plan area. The profile drag coefficient of a flat plate is 1.28 [24].

The weights of the powerplant, structures, and systems can be approximated based on the weight of MAVs in its class, excluding the weight of the batteries, as the Solar Swifter is designed to be entirely solar-powered at this stage. For MAVs in this class, the weight of batteries and/or solar panels can constitute up to 40% of the maximum take-off weight [10, 25, 4]. For instance, a solar-powered quadcopter described in [19] weighs 70 g, with the weight excluding batteries and solar panels being 50 g. This information enables a preliminary power budget estimation. The power required for vertical flight can be evaluated using the following equation.

$$P_{VTOL} = \frac{1}{\eta_p} V_c (50\times 10^{-3}\times 9.81 + 0.0458S + \frac{1}{2}\rho V_c^2 S C_{D_P}) \quad (3)$$

η_p is the propulsive efficiency taken to be 80% and the profile drag coefficient is taken to be 1.28.

Similarly for the forward flight mode the power required is given by

$$P_{FF} = \frac{1}{\eta_p}\frac{1}{2}\rho V^3 S (C_{D_0} + \frac{1}{\pi A R e} {C_L}^2) \quad (4)$$

The 10 illustrates the flight capabilities of the conceptual design. For an aspect ratio of 2, which corresponds to a rectangular wing, the available power is 1.9 W. This configuration supports a cruise velocity of $8m/s$ and a maximum vertical climb velocity of $2.5m/s$. In contrast, an aspect ratio of 1, corresponding to a square-shaped wing, significantly enhances performance. With this configuration, the available power increases to 3.7 W, enabling a cruise velocity of 9 m/s and a vertical climb velocity of 4.2 m/s.

Furthermore, for the given surface area the addition of ultra

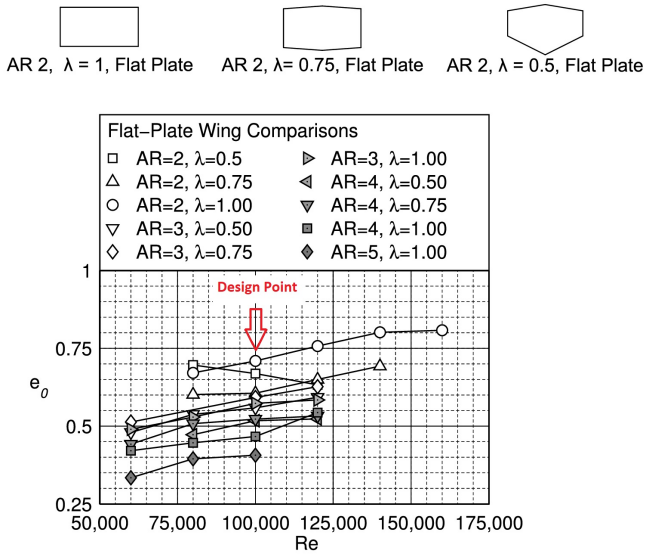


Figure 9: The effect of aspect ratio (AR) and taper ratio (λ) on Oswald's efficiency for different Reynolds number (derived from [23].

thin OPV solar panels add less than a gram of weight which is less than two percent of the take-off weight. This extraordinary power to weight ratio may allow open up the possibility of increasing the payload performance.

7 Conclusion and Future Work

The preliminary analysis have shown that with the recent development of ultrahin OPV solar panels a solar powered flight of the proposed configuration is possible. The performance is predominantly dependent upon the aspect ratio, a lower aspect ratio $AR < 2$ implies high surface area and Oswald efficiency factor. At the scale of six inch span a maximum cruise velocity of 9 m/s and a vertical climb velocity of 4.4 m/s is possible.

The preliminary performance analysis of the Solar Swifter serves as a foundation for the future development of renewable energy-powered hybrid MAVs. Further exploration is needed to investigate the potential for integrating vertical and forward thrusters, which could reduce stall velocity and expand the range of operational velocities. This will involve examining the impact of fan-in-wing configurations on the MAV's aerodynamic performance during forward flight. While the effects of fan-in-wing designs have been studied at higher Reynolds numbers [20], no studies currently exist for low Reynolds numbers (i.e. $Re < 1 \times 10^5$).

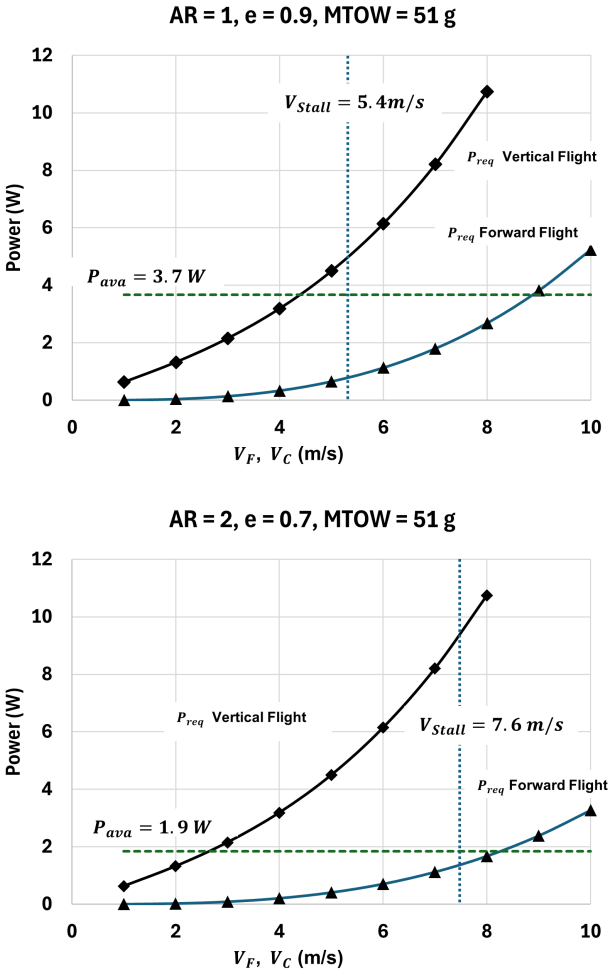


Figure 10: Preliminary power budgeting for forward and vertical flight.

References

[1] M.H. Sabour, P. Jafary, and S. Nematiyan. Applications and classifications of unmanned aerial vehicles: A literature review with focus on multi-rotors. *The Aeronautical Journal*, 2022.

[2] R Yousaf, A Shahzad, MN Mumtaz Qadri, and A Javed. Recent advancements in flapping mechanism and wing design of micro aerial vehicles. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 2020.

[3] Miao Yang Yuhe Zheng Shi Yin Chengjian Hong Fuguo Peng Minghao Qu Chaowei Xue Junxiong Lu Liang Fang Chao Su Daifen Chen Junhua Xu Chao Yan Zhenguo Li Xixiang Xu Zongping Shao Yang Li, Xiaoning Ru. Flexible silicon solar cells with high power-to-weight ratios. *Nature*, 2024.

[4] M. Vaughan C. Roberts and W. J. Bowman. Development of a solar powered micro air vehicle. In *40 Aerospace Sciences Meeting and Exhibit*, 2002.

[5] Martin Kaltenbrunner, Getachew Adam, Eric Daniel Głowacki, Michael Drack1, Reinhard Schwödiauer, Lu-

http://www.imavs.org/

cia Leonat, Dogukan Hazar Apaydin, Heiko Groiss, Markus Clark Scharber, Matthew Schuette White, Niyazi Serdar Sariciftci, and Siegfried Bauer1. Flexible high power-per-weight perovskite solar cells with chromium oxide–metal contacts for improved stability in air. *Nature Materials*, 2015.

[6] Sixing Xiong, Kenjiro Fukuda, Shinyoung Lee, Kyohei Nakano, Xinyun Dong, Tomoyuki Yokota, Keisuke Tajima, Yinhua Zhou, and Takao Someya. Ultrathin and efficient organic photovoltaics with enhanced air stability by suppression of zinc element diffusion. *Advance Science*, 2022.

[7] Winslow Justin, Otsuka Hikaru, Govindarajan Bharath, and Chopra Inderjit. Basic understanding of airfoil characteristics at low reynolds numbers (104–105). *Journal of Aircraft*, 55(3), 2018.

[8] J.H.Han, Y.J. Han., H. H. Yang, S. G. Lee, and E. H. Lee. A review of flapping mechanisms for avian-inspired flapping-wing air vehicles. *Aerospace*, 2023.

[9] DJI Mavic 3 Pro. Dji mavic 3 pro specification. In *https://www.dji.com/uk/mavic-3-pro/specs*, 2024.

[10] Matthew Keennon and Joel Grasmeyer. Development of two mavs and vision of the future of mav design. In *AIAA International Air and Space Symposium and Exposition: The Next 100 Years*, 2003.

[11] M. Yan and T. Ebel. *Micro Air Vehicle: Chapter 5 - Titanium and titanium alloys in drones and other small flying objects*. ELSEVEIR, 2019.

[12] Robert J. Boucher. Sunrise, the world's first solar-powered airplane. *Journal of Aircraft*, 2012.

[13] Andre Noth. History of solar flight.

[14] Ernst Schoeberl. From sunrise to solar impulse 34 years of solar powered flight. In *Presented at the XXIX OSTIV Congress, Lusse, Germany*, 2008.

[15] Andrew Blakers. Development of the perc solar cell. *IEEE JOURNAL OF PHOTOVOLTAICS*, 2019.

[16] Shen Kai, Li Qiang, Wang Deliang, Yang Ruilong, Deng Yi, and Jeng Ming-Jer.

[17] Jianqi Zhang Tao Zhang Yuming Wang Ling Hong Kaihu Xian Bowei Xu Shaoqing Zhang Jing Peng Zhixiang Wei Feng Gao Jianhui Hou Yong Cui, Huifeng Yao. Over 16*Nature Communications*, 2015.

[18] H Ross. Fly around the world with a solar powered airplane. In *In: Proceedings of the 26th congress of international council of the aeronautical science AIAA, Anchorage, Alaska*, 2008.

[19] Aly Abidali1, StephenA.Agha, Antonio Munjiza1, and Mohammad H. Shaheed. Development of a solar powered multirotor micro aerial vehicle. *Nature Scientific Reports*, 2024.

[20] N. Thouault, C. Gologan, C. Breitsamter, and N.A. Adams*. Aerodynamic investigations on a generic fan-in-wing configuration. In *26TH INTERNATIONAL CONGRESS OF THE AERONAUTICAL SCIENCES*, 2008.

[21] J. Hall and k Mohseni. *Encyclopedia of Microfluidics and Nanofluidics.*, chapter Micro Aerial Vehicles. Springer Science, 2015.

[22] F. M. White. *Fluid Mechanics*. McGraw-Hill Education, 2011.

[23] G.K. Ananda, P.P. Sukumar, and M.S. Selig. Measured aerodynamic characteristics of wings at low reynolds numbers. *Aerospace Science and Technology*, 2014.

[24] B. Saeed and G.B. Gratton. An evaluation of the historical issues associated with achieving non-helicopter v/stol capability and the search for the flying car. *The Aeronautical Journal*, 2010.

[25] Luca Petricca, Per Ohlckers, and Christopher Grinde. Micro- and nano-air vehicles: State of the art. *International Journal of Aerospace Engineering*, 2011.

http://www.imavs.org/

A nonlinear autopilot for a piloted Quad-plane

D. A. Martínez-Velasco*, M. A. Martínez-Ramírez*, and H. Rodríguez-Cortés†
*Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional,
Sección de Mecatrónica, Depto. de Ingeniería Eléctrica,
Ciudad de México, 07360, México
†Instituto Tecnológico Autónomo de México,
Campus Río Hondo, Departamento de Ingeniería Eléctrica y Electrónica,
Río Hondo 1, Ciudad de México, 01080, México

http://www.imavs.org/

Abstract

Europe expects to realize urban air mobility (UAM) in five years. The initial commercial flights may require an onboard pilot, but eventually, the operations will be fully autonomous. Hybrid aerial vehicles that combine fixed and rotary wing capabilities, such as the Quad-plane, are emerging as the leading option for UAM. The Quad-plane combines a quadrotor with a fixed-wing configuration. This article proposes a nonlinear autopilot to simplify the pilot workload in different phases of the aircraft flight envelope. The nonlinear autopilot combines various nonlinear control techniques integrated using a state machine. During take-off and landing, in a quadrotor configuration, the pilot controls the aircraft's Cartesian position while the autopilot stabilizes the attitude and height. The autopilot also manages the transitions from quadrotor to fixed-wing aircraft and vice versa. While operating as a fixed-wing aircraft, the auto- pilot commands the aircraft's aerodynamic speed, the flight path angle, and the lateral dynamic. The autopilot's performance is evaluated using Software in the Loop (STIL) with a co-simulation with MATLAB-Simulink and the X-Plane flight simulator.

1 Introduction

In recent years, passenger transportation in metropolitan areas has demanded solutions that increase the quality of life and satisfy ecological constraints. The current technological improvements in electrical propulsion, digitalized air traffic control, and flight control architectures are enabling the potential introduction of electric vehicles that combine vertical take-off and landing capabilities with the advantages of fixed-wing configurations to propose innovative and sustainable urban air mobility solutions [1, 2].

The two most prevalent aircraft configurations in aircraft design are the fixed-wing and rotary-wing configurations; their advantages and disadvantages are well-studied and known. Fixed-wing configurations boast higher cruise speed, endurance, and range than their counterparts. Additionally, they can carry heavier payloads. However, the main drawback lies in the fact that fixed-wing aircraft need a runway for take-off and landing, posing a significant challenge for urban air mobility solutions. On the other hand, rotatory-wing aircraft have vertical take-off and landing capabilities and the ability to hover. Nevertheless, they suffer from lower cruise speed, endurance, and range than their fixed-wing counterparts. Therefore, recently, aircraft designers have been exploring new configurations aiming to combine the advantages of both basic configurations, resulting in new conceptual designs such as hybrid aircraft.

The leading hybrid aircraft configurations are tail-sitter, tail-rotor, and tail-wing. In these configurations, the whole aircraft, the rotors, or the wing are rotated around the aircraft's lateral axis to align the power plant with the vertical or longitudinal axes according to the flight condition [3]. Recently, a new hybrid configuration, the Quad-plane, has been proposed. It is a straightforward combination of a quadrotor and a fixed-wing aircraft. The main difference between a Quad-plane and the other hybrid aircraft configurations, tail-sitter, tilt-rotor, and tilt-wing, is that the Quad-plane has a rotor dedicated to the fixed-wing aircraft configuration. As a result, the Quad-plane has redundant control inputs in some sectors of its flight envelope but is mechanically the simplest among the other hybrid configurations.

The Quad-plane configuration has a greater flight envelope, including VTOL capability and high cruise speeds; for that reason, this configuration has become an attractive alternative for multiple applications, whether

*E-mail: {dalexis.martinez,marcoa.martinez}@cinvestav.mx
†E-mail: hugo.rodriguez@itam.mx On sabbatical leave from Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, Sección de Mecatrónica, Depto. de Ingeniería Eléctrica
** The second and third authors thank the support of Mexican CONAHCYT under project Ciencia de Frontera CF-2023-I-551

http://www.imavs.org/

civil or military, and a strong contender to provide air urban mobility solutions. However, the tail-sitter aircraft from [4] has proved to be efficient in many civil applications.

Hybrid configurations have captured the scientific community's attention due to the complexity of the transition from one configuration to the other [5], [6]. During the transition, the aircraft has over-actuated degrees of freedom, requiring specific control inputs to avoid undesirable behavior. In particular, along the transition from fixed-wing to rotatory-wing configurations, it is essential that the control system for both configurations work in a coordinated way to avoid the wing's stall that could create a descending acceleration that the rotatory-wing configuration could not compensate for. Consequently, flight control design for Quad-planes is an active research subject. In reference [7], a robust nonlinear controller based on the incremental nonlinear dynamic inversion (INDI) approach so that the Quad-plane can follow a given path autonomously. The controller is complemented with reference models to the inner loops to shape the input commands and provide feedforward inputs. Numerical simulations are presented to evaluate the controller's performance. The work reported in [8] shows the design, dynamic modeling, and control synthesis for a QuadPlane. For each flight mode, there is a control algorithm; a four state hybrid automaton coordinates the transitions among the control algorithms. This control algorithm is validated through numerical simulations. In [9], a model-based nonlinear weight assignment principle and an active disturbance rejection control are proposed to improve the altitude control performance during the transition phase. The effectiveness of the aerodynamic model and controller is verified via simulations. In [10], based on the sliding mode control technique, an energy-effective altitude control law for a pusher-type QuadPlane UAV in transition phases is reported. The proposed control law estimates the aerodynamic forces along the longitudinal axis to optimize the required energy to hold the aircraft at a desired altitude. Simulation and actual flight test results show the proposed control performance. In [11], a super twisting sliding mode controller is designed to control the Quad-plane altitude during transition phases. The proposed algorithm is tested using actual UAV's aerodynamic parameters in the X-plane 11 simulation environment. In [12], a controller for all flight phases in a Quad-Plane is reported. The controller is based on the Incremental Nonlinear Control Allocation (INCA) technique. This controller avoids negative wing angles of attack, minimizing the loads on the rotors. Numerical simulations and experimental flights are presented to validate the algorithm. In [13], an observer-based optimal control approach for active combined fault and wind disturbance rejection for a Quad-plane. The control synthesis considers a linear model for the longitudinal plane and the transition modes. A numerical simulation demonstrates that for a typical Quad-plane flight profile at 100 m altitude, the observer-based wind gust and fault correction significantly enhance trajectory tracking performance.

Existing literature offers limited insights into the nonlinear nature of the Quad-plane dynamics and its full flight envelope. The framework presented in this study fills these gaps by designing the controllers considering the nonlinear Quad-plane model and covering a full flight mission from take-off to landing, going through the transition from quadrotor to fixed-wing aircraft and viceversa. This novel approach is a significant step toward understanding the Quad-plane capabilities.

Hence, this work introduces a nonlinear autopilot for different phases of a Quad-plane flight envelope, including the crucial transition from rotatory to fixed-wing configuration and vice versa. The autopilot is based on various control techniques and includes an exogenous disturbance estimator based on immersion and invariance [14]. For take-off and landing, the nonlinear autopilot commands the quadrotor attitude around the lateral and longitudinal axes, the angular speed around the vertical axis, and the height. The pilot controls the Cartesian position. For the transition mode from quadrotor to fixed-wing, the fixed-wing power plant accelerates the vehicle until the longitudinal speed is bigger than the wing stall speed; in this flight phase, the autopilot commands the vehicle's attitude using the auditor power plant. Once this condition on the longitudinal speed is satisfied, a longitudinal autopilot based on the Total Energy Control System approach commands the aircraft's aerodynamic speed. At the same time, the pilot can control the aircraft's translational position. This work is a direct continuation of the preliminary report in [15].

The outline of this report is as follows. Section 2 presents the QuadPlane dynamic model that will be taken into consideration for the control algorithms design. Section 3 addressed the control algorithms for each flight phase. Section 4 reports state machine that mixes each algorithm in order to achieve a smooth transition from pure mode flights. Section 5 presents the numerical simulation results obtained using the flight simulator X-Plane. This work ends at section 6 with some remarks and conclusions.

2 Quad-Plane dynamic model

The critical flight phase for a Quad-Plane aircraft happens at the transition from fixed-wing to rotatory-wing configurations and vice versa. Since the aircraft must hold a speed higher than the wing stall speed in a fixed-wing configuration, the transition's main features occur on the longitudinal plane. However, the whole

http://www.imavs.org/

rotational dynamic must be considered since the lateral movements and rotation around the vertical axis must be kept bounded.

The Quad-plane model considered in this work is shown in Figure 1. The aircraft has four rotors as the power plant for the quadrotor configuration r_i, $i = 1, , 4$, and a fifth rotor r_5 for the fixed-wing configuration. Moreover, the vehicle has ailerons, elevators, and rudder for fixed-wing operation.

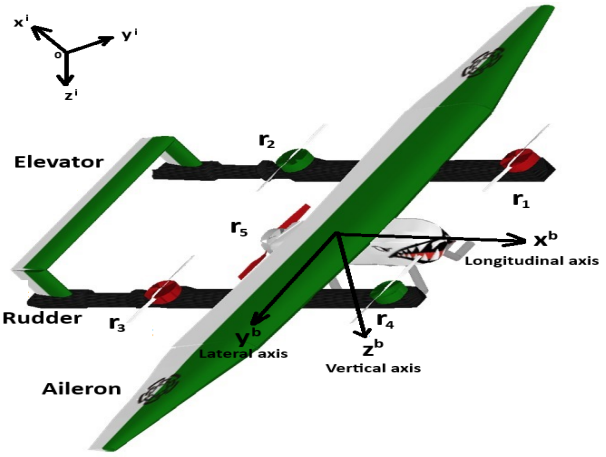


Figure 1: QuadPlane model. Body coordinated axis system $0x^by^bz^b$ and inertial coordinated axis system $0x^iy^iz^i$

The following differential equations describe the Quad-plane longitudinal translational dynamic model in mixed body and inertial coordinates [16]

$$\begin{aligned} \dot{z} &= -us_\theta + wc_\theta \\ m\dot{u} &= -mqw - mgs_\theta + \mathbf{q}SC_x + \kappa mg\delta_T \\ m\dot{w} &= mqu + mgc_\theta + \mathbf{q}SC_z - T_T \end{aligned} \tag{1}$$

with z the aircraft inertial position along the $0z^i$ inertial axis, u and w are the velocities along the $0x^b$ and $0z^b$ axes, respectievly. θ is the pitch angle, m the Quad-plane mass, q the pitch angular speed, g the gravity constant,

$$\mathbf{q} = \frac{1}{2}\rho(u^2 + w^2)$$

the dynamic pressure, with ρ the air density, S the Quad-plane wing surface. Moreover, $\delta_T \in (0, 1)$ the accelerator position for the fifth rotor, and $T_T = \sum_{i=1}^{4} T_i$ the total thrust generated by the quadrotor' rotors. Finally, C_x and C_z are the aerodynamic force coefficients.

The rotational dynamic model is given by

$$\begin{aligned} \dot{R} &= R\,\mathbf{S}(\Omega) \\ J\dot{\Omega} &= -\mathbf{S}(\Omega)J\Omega + M_P^b + M_Q^b \end{aligned} \tag{2}$$

with $R \in SO(3)$ the rotation matrix, with

$$SO(3) = \{R \in \mathbb{R}^{3\times3} | R^\top R = I, \det(R) = 1\}$$

The map $\mathbf{S}(\cdot) : \mathbb{R}^3 \to \mathfrak{so}(3)$, with $\mathfrak{so}(3)$ the Lie algebra of $SO(3)$, is defined as $a \times b = \mathbf{S}(a)b$ for all vectors $a, b \in \mathbb{R}^3$. Moreover, $\Omega = \begin{bmatrix} p & q & r \end{bmatrix}^\top$ is the angular velocity in the body coordinates, and

$$M_P^b = \mathbf{q}S \begin{bmatrix} b(C_l(\cdot) + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r) \\ \bar{c}(C_m(\cdot) + C_{m_{\delta_e}}\delta_e) \\ b(C_n(\cdot) + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r) \end{bmatrix}$$

are the aerodynamic moments[1] in fixed-wing configuration, where b is the wingspan, $C_l(\cdot)$ the roll aerodynamic moment coefficient, $C_{l_{\delta_a}}$ and $C_{l_{\delta_r}}$ the roll and yaw aerodynamic control coefficients, respectively, δ_a and δ_r the aileron and rudder deflection angles, respectively, $\bar{c}$ is the wing mean aerodynamic chord, $C_m(\cdot)$ the pitch aerodynamic moment coefficient, $C_{m_{\delta_e}}$ the pitch aerodynamic control coefficient, δ_e the elevator deflection angle, $C_n(\cdot)$ the yaw aerodynamic moment coefficient, $C_{n_{\delta_a}}$ and $C_{d_{\delta_r}}$ the roll and yaw aerodynamic control coefficients for yaw, respectively. Finally,

$$M_Q^b = \begin{bmatrix} M_{Q_p}^b \\ M_{Q_q}^b \\ M_{Q_r}^b \end{bmatrix} = \begin{bmatrix} l_1(T_1 + T_4 - T_3 - T_2) \\ l_2(T_1 + T_2 - T_3 - T_4) \\ Q_1 + Q_3 - Q_2 - Q_4 \end{bmatrix}$$

are the applied moments in quadrotor configuration with l_1 and l_2 the distances from the $0x^b$, $0y^b$ axes to the rotors' position. $T_i, i = 1, 2, 3, 4$ the thrust generated by each rotor and $Q_i, i = 1, 2, 3, 4$ the counter-rotating moment produced by each rotor.

Note that in both configurations, the rotational dynamic is fully actuated. However, the control moments from the aerodynamic surfaces are effective only after the Quad-plane aerodynamic speed is higher than the stall speed; below the wing stall speed, these aerodynamic moments act as disturbances for the quadrotor configuration. Conversely, the control inputs for the quadrotor configuration and the fifth rotor are available for the entire flight envelope.

3 Control Algorithms.

This section presents the control algorithms that are used during the transition phase from rotatory-wing to fixed-wing and vice versa. The control algorithms were developed using different control techniques and disturbances estimators making use of immersion and invariance.

[1] $(\cdot)$ represents the aerodynamic coefficients dependency on the aerodynamic angles, the angle of attack and the sideslip angle, the Reynolds number, and the angular velocity [16].

http://www.imavs.org/

3.1 Attitude autopilot in quadrotor configuration (AAQC)

The attitude autopilot in quadrotor configuration works during take-off and landing procedures. Thus, the objective of this autopilot is to command the Quad-plane angular position around the lateral and longitudinal axes and the angular velocity around the vertical axis. The pilot can modify the angular and angular speed references to command the vehicle's position in the Cartesian plane.

The rotation matrix R is parameterized using the Tait-Bryan angles to design the AAQC autopilot

$$R = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\theta s_\psi & c_\phi s_\theta c_\psi + s_\theta s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\theta c_\psi & c_\phi s_\theta s_\psi - s_\theta c_\psi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix}$$

with ϕ the roll and yaw angles, respectively. By introducing the following notation for the columns of the matrix R

$$R = \begin{bmatrix} r_1^\top & r_2^\top & r_3^\top \end{bmatrix}$$

the first equation from (1) can be written in the following form [17]

$$\dot{r}_1 = r_1 \times \Omega, \quad \dot{r}_2 = r_2 \times \Omega, \quad \dot{r}_3 = r_3 \times \Omega$$

On the other hand, note that the gravity force acting on the vehicle expressed in body axes reads as

$$F_g^b = R^\top F_g^i = mgr_3$$

where $F_g^i = \begin{bmatrix} 0 & 0 & mg \end{bmatrix}^\top$. Hence, the AAQC objective is stated as follows: design control inputs M_Q^b to align the vector r_3 with $e_z = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$ and to keep $r = 0$ for take off and landing and to $r_{3_d} = \begin{bmatrix} -s_{\theta_d} & c_{\theta_d} s_{\phi_d} & c_{\theta_d} c_{\phi_d} \end{bmatrix}$ and $r_d = 0$ for pilot commands.

The autopilot is designed following the nonlinear approach proposed in [18]. Thus, the attitude error is defined as $\tilde{r}_3 = r_3 \times r_{3_d}$. Then, one obtains

$$\dot{\tilde{r}}_3 = -\mathbf{S}(\Omega_d)\tilde{r}_3 - \mathbf{S}(r_{3_d})S(r_3)\tilde{\Omega} \tag{3}$$

with $\tilde{\Omega} = \Omega - \Omega_d$. Assuming that the references send by the pilot are piece wise constant, it is considered that $\dot{r}_{3_d} = 0$. The reference for the angular speed sent by the pilot is considered by defining $\Omega_d = \begin{bmatrix} 0 & 0 & r_d \end{bmatrix}^\top$.

The angular velocity error dynamic is described by

$$\begin{aligned} \dot{\tilde{\Omega}} &= J^{-1}\Big(-\mathbf{S}(\tilde{\Omega})J(\tilde{\Omega} + \Omega_d) - \mathbf{S}(\Omega_d)J\tilde{\Omega} \\ &\quad -\mathbf{S}(\Omega_d)J\tilde{\Omega}_d + M_P^b + M_Q^b\Big) \end{aligned} \tag{4}$$

with r_d assumed piece wise constant.

It is important to note that under low values of velocity, M_P^b takes values close to zero, so it can be considered a disturbance. Hence, for control design, the dynamic model (4) is expressed as

$$J\dot{\tilde{\Omega}} = -\mathbf{S}(\tilde{\Omega})J(\tilde{\Omega} + \Omega_d) - \mathbf{S}(\Omega_d)J\tilde{\Omega} + M_Q^b + \delta_M \tag{5}$$

with $\delta_M = -\mathbf{S}(\Omega_d)J\tilde{\Omega}_d + M_P^b$ modeling a disturbance with upper bounded first time derivative, this is, $||\dot{\delta}_M|| \leq k_0$ for some constant k_0.

The following controller is proposed

$$M_Q^b = -K_P\tilde{r}_3 - K_D\tilde{\Omega} - \hat{\delta}_M - \Gamma J\tilde{\Omega} \tag{6}$$

with K_P, K_D and Γ positive defined matrices. The error dynamics (3)-(5) in closed-loop with (6) reads as

$$\begin{aligned} \dot{\tilde{r}}_3 &= -S(\Omega_d)\tilde{r}_3 - S(r_{3_d})S(r_3)\tilde{\Omega} \\ J\dot{\tilde{\Omega}} &= -\mathbf{S}(\tilde{\Omega})J(\tilde{\Omega} + \Omega_d) - \mathbf{S}(\Omega_d)J\tilde{\Omega} - K_P\tilde{r}_3 \\ &\quad -K_D\tilde{\Omega} + \tilde{\delta}_M \\ \dot{\tilde{\delta}}_M &= -\Gamma\tilde{\delta}_M + \dot{\delta}_M \end{aligned} \tag{7}$$

where

$$\begin{aligned} \tilde{\delta}_M &= \delta_M - \hat{\delta}_M - \Gamma J\tilde{\Omega} \\ \dot{\tilde{\delta}}_M &= -\Gamma\Big(-\mathbf{S}(\tilde{\Omega})J(\tilde{\Omega} + \Omega_d) - \mathbf{S}(\Omega_d)J\tilde{\Omega} + \hat{\delta}_M \\ &\quad +\Gamma J\tilde{\Omega} + M_Q^b\Big) \end{aligned} \tag{8}$$

with $\tilde{M}$ the disturbance estimation error and $\hat{M}$ the state of the disturbance estimator.

3.2 Height Autopilot in Quadrotor Configuration (HAQC)

The HAQC aims to command the vehicle's height during take-off and landing operations without vertical speed measurements. The HAQC uses the quadrotor power plant to achieve this objective. For autopilot design, the following model is considered

$$\begin{aligned} \dot{\tilde{h}} &= -wc_\theta \\ \dot{w} &= gc_\theta - \tfrac{T_T}{m} + \delta_w \end{aligned} \tag{9}$$

with $\tilde{h} = h - h_d$ the heighterror, h the actual height and h_d the desired constant height. $\delta_w = qu + \mathbf{q}SC_z/m$ represents a disturbance. Once again, it is assumed that $|\dot{\delta}_z| \leq \kappa_1$ for some constant κ_1. The following control law is proposed

$$T_T = m(-k_p\tilde{h} + k_d(\hat{w} - \gamma_1\tilde{h}) - \gamma_2(\hat{w} - \gamma_1\tilde{h}) + \hat{\delta}_w) \tag{10}$$

where k_p, k_d, γ_1 and γ_2 are the autopilot gains, and

$$\begin{aligned} \dot{\hat{w}} &= gc_\theta + \hat{\delta}_w - \gamma_2(\hat{w} - \gamma_1\tilde{h}) - T_T/m \\ &\quad +\gamma_1(\hat{w} - \gamma_1\tilde{h})c_\theta \\ \dot{\hat{\delta}}_w &= \gamma_2(gc_\theta + \hat{\delta}_w - \gamma_2(\hat{w} - \gamma_1\tilde{h}) - T_T/m) \end{aligned} \tag{11}$$

where $\hat{w}$ is the vertical observer state, $\hat{\delta}_w$ is the state of the disturbance estimator. This observer/estimator is designed following the procedure reported in [19].

The system (9) in closed-loop with (10) and (11) is descrbed by

$$\begin{array}{rcl} \dot{\tilde{h}} & = & -wc_\theta \\ \dot{w} & = & k_p\tilde{h} - k_d w + k_d\tilde{w} + \tilde{\delta}_w \\ \dot{\tilde{w}} & = & \tilde{\delta}_w + \gamma_1\tilde{w}c_\theta \\ \dot{\tilde{\delta}}_w & = & -\gamma_1\gamma_2\tilde{w}c_\theta + \dot{\delta}_w \end{array} \tag{12}$$

where

$$\begin{array}{rcl} \tilde{w} & = & w - \hat{w} + \gamma_1\tilde{h} \\ \tilde{\delta}_w & = & \delta_w - \hat{\delta}_w + \gamma_2(\hat{w} - \gamma_1\tilde{h}) \end{array}$$

are the vertical speed observer and the estimation errors, respectively.

3.3 Longitudinal Velocity Autopilot in Quadrotor Configuration (LVAQC)

The LVAQC works only during the transition from quadrotor to fixed-wing. The desired velocity is set as the wing stall speed, and the fifth rotor is employed to attain it. In this flight state, the aircraft could accelerate longitudinally using the quadrotor power plant or the fifth rotor power. Here, it is proposed to use the fifth to accelerate longitudinally and to control the vehicle's attitude using the quadrotor power plant. The autopilot is designed considering the following dynamic model

$$\dot{u} = -gs_\theta + \kappa g\delta_T + \delta_u \tag{13}$$

where δ_T is the accelerator position for the fifth rotor and $\delta_u = qw + \mathbf{q}SC_x$ is considered as disturbance with upper bounded first time derivative. Thus, there exist κ_2 such that $|\delta_u| \leq \kappa_2$.

The following autopilot algorithm is proposed

$$\begin{array}{rcl} \delta_T & = & \frac{1}{\kappa g}(-k_{p_u}\tilde{u} + gs_\theta - \sigma_1 u - \hat{\delta}_u) \\ \dot{\hat{\delta}}_u & = & -\sigma_1(-gs_\theta + \sigma_1 u + \kappa g\delta_T + \hat{\delta}_u) \end{array} \tag{14}$$

with k_{p_u} and γ_1 autopilot positive gains, $\tilde{u} = u - u_d$ the longitudinal error speed, and u_d the reference speed equal to the wing stall speed. The closed-loop dynamics (13)-(14) reads as

$$\begin{array}{rcl} \dot{\tilde{u}} & = & -k_{p_u}\tilde{u} + \tilde{\delta}_u \\ \dot{\tilde{\delta}}_u & = & -\sigma_1\tilde{\delta}_u + \dot{\delta}_u \end{array} \tag{15}$$

3.4 Longitudinal Speed autopilot in fixed-wing configuration (LSAFC)

During the transition from fixed-wing to quadrotor, the aerodynamic speed must decrease to the wing stall speed. At this point, the autopilots AAQC and HAQC are activated. The following dynamic model is considered to design the autopilot engaged to decrease the aircraft's aerodynamic speed

$$\begin{array}{rcl} \frac{\dot{u}}{g} & = & -s_\theta + \kappa\delta_T + \frac{\delta_u}{g} \\ \dot{\theta} & = & q \\ \dot{q} & = & \mathbf{q}\frac{S\bar{c}}{I_{yy}}(C_m(\cdot) + C_{m_{\delta_e}}\delta_e) \end{array} \tag{16}$$

The following autopilot algorithm is proposed

$$\delta_T = \frac{1}{\kappa}\left(-\frac{\hat{\delta}_u + \sigma_1 u}{g} + \bar{\delta}_T\right) \tag{17}$$

with σ_2 a positive gain. Moreover,

$$\begin{array}{rcl} \bar{\delta}_T & = & \frac{1}{1+k_{p_h}}(k_{p_h}(2s_\theta + \frac{k_v}{g}\tilde{u}) + k_{i_h}\eta_1) \\ \dot{\eta}_1 & = & -\bar{\delta}_T + 2s_\theta + \frac{k_v}{g}\tilde{u} \end{array} \tag{18}$$

where k_{p_h}, k_{i_h} and k_v are positive constant gains, and

$$\begin{array}{rcl} \delta_e & = & -k_p s_\theta - k_d q + k_{p_l}(-\bar{\delta}_T + \frac{k_v}{g}\tilde{u}) + k_{i_l}\eta_2 \\ \dot{\eta}_2 & = & -\bar{\delta}_T + \frac{k_v}{g}\tilde{u} \end{array} \tag{19}$$

with k_p, k_d, k_{p_l}, and k_{i_l} positive constant gains. The closed-loop dynamic (16)-(19) is described by the following equations

$$\begin{array}{rcl} \frac{\dot{\tilde{u}}}{g} & = & \left(\frac{2k_{p_h}}{1+K_{p_h}} - 1\right)s_\theta + \frac{k_{p_h}}{1+k_{p_h}}\frac{k_v}{g}\tilde{u} + \frac{\tilde{\delta}_u}{g} \\ \dot{\theta} & = & q \\ \dot{q} & = & \mathbf{q}\frac{S\bar{c}}{I_{yy}}(C_m(\cdot) + C_{m_{\delta_e}}(-k_p s_\theta - k_d q \\ & & +k_{p_l}(-\bar{\delta}_T + \frac{k_v}{g}\tilde{u}) + k_{i_l}\eta_2)) \\ \dot{\eta}_1 & = & -\bar{\delta}_T + 2s_\theta + \frac{k_v}{g}\tilde{u} \\ \dot{\eta}_2 & = & -\bar{\delta}_T \\ \dot{\tilde{\delta}}_u & = & -\sigma_1\tilde{\delta}_u + \dot{\delta}_u \end{array} \tag{20}$$

It can be verified that if $\dot{\delta}_u = 0$ the only equilibrium point in (20) is $\tilde{u} = 0$. Using a linear approximation of the closed-loop dynamic in (20) the gains can be determined to prove local stability. The controller defined in (17)-(19) is based on the Total Energy Control described in [20] for the lineal case and [21] for the non-lineal case.

4 State Machine

This work proposes using a state machine with finite states to allow the pilot to switch among the proposed autopilots. The machine states are defined as [22]

$$\mathcal{Q} = \{\mathbf{LQM}, \mathbf{QM}, \mathbf{TQ2FW}, \mathbf{FWM}, \mathbf{TFW2Q}\}$$

with **LQM** landing in quadrotor configuration mode, **QM** quadrotor operation mode, including take-off,

http://www.imavs.org/

http://www.imavs.org/

TQ2FW transition from quadrotor to fixed-wing configurations, **FWM** fixed-wing operation mode, and **TFW2Q** transition from fixed-wing to quadrotor configurations. The automata's alphabet is defined by the binary position of two switches in the radio transmitter and a comparison between the actual longitudinal velocity and the wing stall longitudinal speed u_d. This is,

$$\sum = \{T_{11}, T_{12}, T_{21}, T_{22}, u > u_d, u < u_d/2\}$$

The machine states diagram in Figure 2 shows the different transitions for the pilot to operate the Quad-plane.

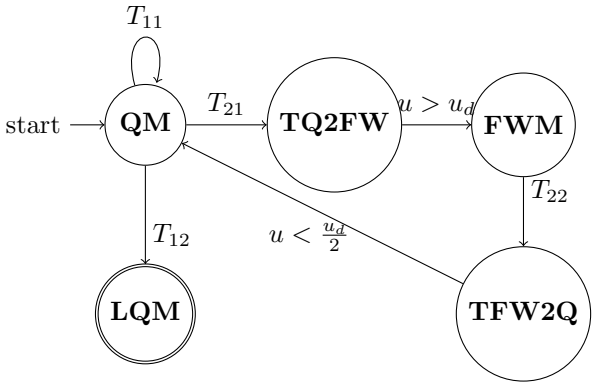


Figure 2: Machine states diagram.

Figure 3 shows the radio transmitter command configuration and the transition switches for take-off and landing. When the states **LQM** or **QM** are selected, *AAQC* controls the aircraft attitude, and the pilot can send roll and pitch angles and the yaw angular speed references; additionally, the *HAQC* regulates the aircraft's height. Once the pilot switches to state **TQ2FW**, the *AAQC*, *HAQC*, and *LVAQC* are active. The pilot can still send references for roll and pitch angles and the yaw angular speed commanded through M_Q^b, but its effectiveness decreases as u increases. In the state **FWM**, the pilot has complete control over the aircraft unless the pilot's thrust command is not giving enough power to reach a minimum speed above the wing stall speed; the LVAQC performs this task. During the **TFW2Q** state is active, the **LSAFC** commands the aircraft's longitudinal speed and the pitch angle. However, as u decreases, the control surfaces lose effectiveness. Hence, once the longitudinal speed is smaller than the wing stall speed, the *AAQC* and *HAQC* take control of the aircraft, and the pilot recovers authority on the vehicle's Cartesian position.



Figure 3: Stick and switch allocation on the transmitter

5 Numerical simulation results

A SITL co-simulation between MATLAB-Simulink and X-Plane [23] is reported to evaluate the autopilot. The X-Plane flight simulator solves the aircraft dynamic model through a simplified CFD (Computational Fluid Dynamic) analysis. The aircraft model was made using Plane-Maker, a software included in X-Plane. The aircraft's main features are reported in Table 1.

Parameter	Value	Parameter	Value
b	2.17m	S	1.32m
m	22.67kg	$\bar{c}$	0.45m
l_1	0.65m	l_2	0.8m

Table 1: Quad-plane physical parameters

The autopilot is evaluated using two simulations. The first considers optimal flight conditions; the second considers light turbulent conditions with wind speeds of about 11kt. In both simulations, the pilot takes off and moves in the Cartesian position in quadrotor mode. Then, the pilot moves the transition switch to the T_{22} position, and the aircraft reaches a longitudinal speed above the wing stall speed. The wing's stall speed was determined following the procedure reported in [24]. Once the state FWM is reached, the pilot has complete control over the aircraft. Then, the pilot selects the switch position T_{21} to start the transition from fixed-wing to quadrotor configuration. Finally, the switch position T_{11} is activated, and the aircraft begins the landing operation, this simulations can be seen at the following link https://youtu.be/c_qRoVBpvrA.

Mode	T_{1i}	T_{2i}
QM,TFW2Q	1	0
LQM	0	1/0
TQ2FW,FWM	1	1

Table 2: Binary channels and the flight modes

http://www.imavs.org/

Figure 4 presents the binary channels′ evolution for the optimal flight conditions simulation. As was described before, the quadrotor takes off as soon as the simulation starts; this is commanded through T_{1i} (solid line) where $T_{11} = 1$ commands the take-off operation and $T_{12} = 0$ commands the landing operation and T_{2i} (segmented line) commands the transition from quadrotor to fixed-wing configuration through $T_{22} = 1$ and the inverse operation through $T_{21} = 0$, the mapping between the binary channels and the flight modes is displayed in Table 2. Figure 5 reports the angular moments calculated by the $AAQC$ around the $0x^b$ and $0y^b$ axes. As expected, these values only differ from zero when $u < u_d$.

Figure 6 shows the longitudinal speed u (left label), the aircraft's height time evolution h (right label), and the Quad-plane flying modes (blue line) in the no turbulence case. As observed, the autopilot ends the transition from quadrotor mode to fixed-wing mode once the aircraft reaches the stall speed (red line), and the transition from fixed-wing to quadrotor mode ends when the aircraft has a longitudinal speed lower than half the stall speed; this is possible since during this transition phase the residual wing's lift and the thrust generated by the rotors compensate the aircraft's weight.

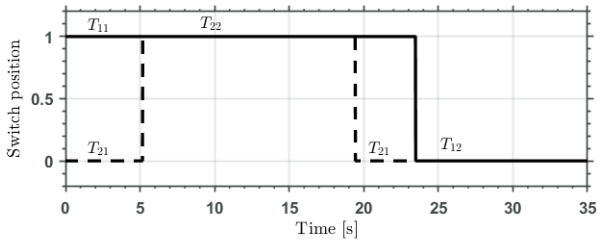


Figure 4: Binary channels evolution. The simulation with no turbulence.

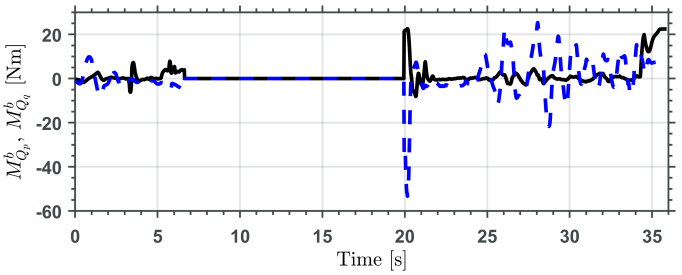


Figure 5: Angular moments computed by $AAQC$. The no turbulence simulation case.

Figure 7 presents the control moments M^b_p and M^b_q generated by the $AAQC$. In this case, these control moments reach higher values than in the previous simulation since the turbulent conditions disturb the aircraft.

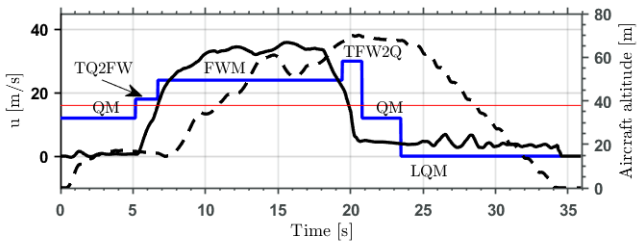


Figure 6: Quad-plane flight mission. Longitudinal speed u (continuos line) – and height h (dashed line). The no turbulence simulation case.

Figure 8 shows the longitudinal speed u (solid line, left label), the aircraft's height time evolution (dashed line, right label) and the Quad-plane transitions (blue line) with turbulence conditions. As a remark, the longitudinal velocity perturbation due to the turbulence conditions can be seen during the landing operation. Also, since the aircraft was flying against the wind, the resulting drag decelerated the vehicle, leading to a really short transition from fixed wing to quadrotor mode ($TFW2Q$), in comparison with the no turbulence case.

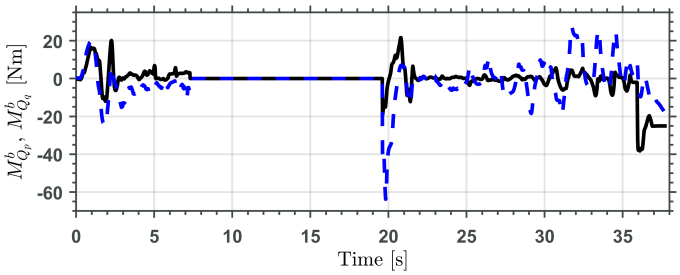


Figure 7: Angular moments calculated by $AAQC$ for the simulation with turbulence.

6 Conclusions

This work proposes an autopilot to assist a pilot in operating a hybrid aircraft known as a Quad-plane. The autopilot comprises four nonlinear control algorithms coordinated through a state machine. The autopilot is evaluated through a SITL co-simulation where the autopilot runs in MATLAB-Simulink while the aircraft dynamic model is simulated in X-Plane. Two simulations are reported. The first has optimal flight conditions, while the second considers turbulent conditions. The turbulence intensity was selected high enough to make

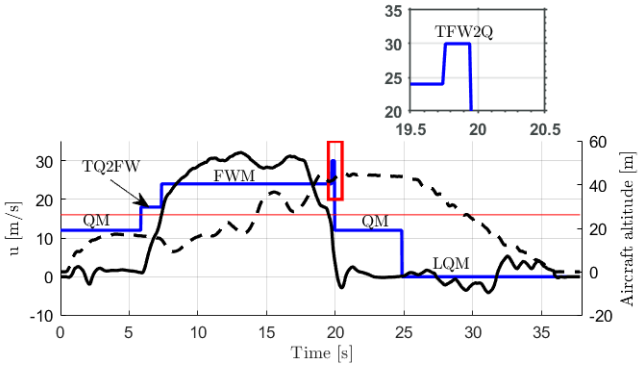


Figure 8: Quad-plane transitions, longitudinal and height speed for the simulation with turbulence.

the flight hard for the pilot. An important issue is left open in this work: the stability analysis of the complete closed-loop dynamic.

References

[1] L. Wang, X. Deng, J. Gui, P. Jiang, F. Zeng, and S. Wan, "A review of urban air mobility-enabled intelligent transportation systems: Mechanisms, applications and challenges," *Journal of Systems Architecture*, p. 102902, 2023.

[2] Q. Long, J. Ma, F. Jiang, and C. J. Webster, "Demand analysis in urban air mobility: A literature review," *Journal of Air Transport Management*, vol. 112, p. 102436, 2023.

[3] G. J. Ducard and M. Allenspach, "Review of designs and flight control techniques of hybrid and convertible vtol uavs," *Aerospace Science and Technology*, vol. 118, p. 107035, 2021.

[4] Wingtra, *A mapping drone for fast and accurate surveying data every time*, Accessed June 2024.

[5] K. Wang, Y. Ke, S. Lai, K. Gong, Y. Tan, and B. M. Chen, "Model-based optimal auto-transition and control synthesis for tail-sitter uav kh-lion," in *2017 13th IEEE International Conference on Control & Automation (ICCA)*, pp. 541–547, 2017.

[6] L. M. Sanchez-Rivera, R. Lozano, and A. Arias-Montano, "Transition flight dynamics of a dual tilt-wing uav," in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 862–866, 2020.

[7] L. Zhou, J. Yang, T. Strampe, and U. Klingauf, "Incremental nonlinear dynamic inversion based path-following control for a hybrid quad-plane unmanned aerial vehicle," *International Journal of Robust and Nonlinear Control*, vol. 33, no. 17, pp. 10304–10327, 2023.

[8] A. Mathur and E. M. Atkins, "Design, modeling and hybrid control of a quadplane," in *AIAA Scitech 2021 Forum*, 2021.

[9] H. Zhao, Y. Xia, D. Ma, C. Hao, and F. Yu, "Active disturbance rejection altitude control for a quadplane," in *Advances in Guidance, Navigation and Control* (L. Yan, H. Duan, and X. Yu, eds.), (Singapore), pp. 329–342, Springer Singapore, 2022.

[10] T. T. Nguyen, D. Van Vu, S. X. Mai, and D. Thanh Nguyen, "An energy-effective altitude controller of a pusher-type hybrid uav in transition phases," in *2022 International Electrical Engineering Congress (iEECON)*, pp. 1–4, 2022.

[11] D. Van Vu, T. T. Nguyen, S. X. Mai, and D. T. Nguyen, "A robust transition control law for a quadplane," in *2022 International Conference on Electronics, Information, and Communication (ICEIC)*, pp. 1–4, 2022.

[12] H. Karssies and C. D. Wagter, "Extended incremental non-linear control allocation (xinca) for quadplanes," *International Journal of Micro Air Vehicles*, vol. 14, p. 17568293211070825, 2022.

[13] Z. Zyadat, N. Horri, M. Innocente, and T. Statheros, "Observer-based optimal control of a quadplane with active wind disturbance and actuator fault rejection," *Sensors*, vol. 23, no. 4, 2023.

[14] A. Astolfi, D. Karagiannis, and R. Ortega, *Nonlinear and Adaptive Control with Applications*, vol. 187. Springer, 2008.

[15] D. A. Martinez-Velasco, M. A. Martinez-Ramirez, and H. Rodriguez-Cortes, "Autopilot for flight mode transistion in quadplanes," *Memorias del Congreso Nacional de Control Automático*, vol. 6, pp. 568–573, 2023.

[16] E. A. Morelli and V. Klein, *Aircraft system identification: theory and practice*, vol. 2. Sunflyte Enterprises Williamsburg, VA, 2016.

[17] T. Lee, M. Leok, and N. H. McClamroch, "Global formulations of lagrangian and hamiltonian dynamics on manifolds," *Springer*, vol. 13, p. 31, 2017.

[18] E. M. Coates and T. I. Fossen, "Geometric reduced-attitude control of fixed-wing uavs," *Applied Sciences*, vol. 11, no. 7, 2021.

[19] J. L. Mendoza-Soto, J. J. Corona-Sánchez, and H. Rodríguez-Cortés, "Quadcopter path following control. a maneuvering approach," *Journal of Intelligent & Robotic Systems*, vol. 93, pp. 73–84, 2019.

[20] A. Lambregts, "Functional integration of vertical flight path and speed control using energy principles," *NASA. Langley Research Center NASA Aircraft Controls Research, 1983*, 1984.

[21] J. Brigido-González and H. Rodríguez-Cortés, "Experimental validation of an adaptive total energy control system strategy for the longitudinal dynamics of a fixed-wing aircraft," *Journal of Aerospace Engineering*, vol. 29, no. 1, p. 04015024, 2016.

[22] M. Sipser, *Introduction to the Theory of Computation*, vol. 27. ACM New York, NY, USA, 1996.

[23] P. Thomas, *X-Plane Blockset*. MATLAB Central File Exchange, Retrieved February 25, 2022.

[24] N. H. McClamroch, *Steady Aircraft Flight and Performance*. Princeton University Press, 2011.

http://www.imavs.org/

Spherical Aerial Manipulator Robot for Exploration in Complex Forest Environments

J. Daniel Olivares-Figueroa, Israel Cruz-Vega *, Alejandro Gutierrez-Giles, and Jose Martinez-Carranza
Insituto Nacional de Astrofisica, Optica y Electronica - Puebla, Mexico

http://www.imavs.org/

ABSTRACT

One significant research area that has greatly benefited from UAVs is precision agriculture, due to their ability to offer excellent spatial resolution, making them ideal for detailed minor-scale analyses of vegetable patches. To the best of our knowledge, few studies have applied UAVs to explore complex forest environments, such as orchards, often relying on indirect methods to obtain crop information. In this work, we propose a novel approach that enables the direct measurement of characteristics, such as individually measuring fruits or stems/leaves, to assess their maturity or detect crop diseases, respectively. To achieve this, we introduce a new design called *"SAMBot: Spherical Aerial Manipulator Robot"*, which consists of a mini-UAV protected by a spherical structure with an attached manipulator at its front. SAMBot combines the protective benefits of the spherical design with the manipulator's ability to access confined spaces. The proposed design is compatible with ROS2, a widely-used framework in robotics research and industrial applications. The range of potential applications for our proposed design extends beyond precision agriculture, encompassing areas such as search and rescue operations in mines or collapsed buildings, structural inspections, and autonomous underground navigation, among others.

1 INTRODUCTION

Unmanned Aerial Vehicles (UAVs), or drones, are aeronautical complex systems that can be controlled without an onboard pilot for autonomous flights. Due to the integration of advanced navigation systems with cameras and sensors, it is possible to collect data for various tasks such as space, sea, and land exploration, as well as numerous applications in the entertainment and military industries, among others [1, 2].

Precision Agriculture (PA) is one of the scientific fields that has most benefited from UAV technology. The increase in the world population and the extreme weather events caused by global warming are making food production increasingly challenging [3]. Drones offer a potential solution to these challenges by improving agricultural efficiency and productivity through crop monitoring, irrigation management, pesticide and fertilizer spraying, cartography, and more [4, 5].

While drone research in precision agriculture has led to significant advancements, it is essential to acknowledge the current limitations. Most studies primarily focus on acquiring images and geospatial data to estimate overall crop health. Drones are restricted to flying over orchards at a specific altitudes to capture thermal, multispectral, or RGB images While effective for global data estimation, this approach does not facilitate the acquisition of local data. In [6], the exploration of a drone through an apple orchard is presented, using an indirect technique to determine fruit maturity by detecting ethylene in the air. In [7], a UAM is used to install and recover sensors in trees, preventing fires through the real-time monitoring. The manipulator has a single prismatic joint for surface contact, which limits its accuracy for other applications. In [8], a UAV installs a sensor network in forests to measure temperature, pressure, and luminosity. The authors do not incorporate a manipulator; instead, the drone is positioned 3 meters from the target to launches the sensors. In [9], a UAM with two passive rotating hemispherical shells is presented for manipulating objects in complex environments during disasters. However, the manipulator is mounted on the drone body, has a single degree of freedom (DoF), and only extends up to 70 mm. We have not found a satisfactory solution that allows for the direct measurement of orchard characteristics, such as assessing individual fruits or stems/leaves to determine their maturity or detect crop diseases.

The importance of obtaining precise information on the ripeness state of each fruit lies in the ability to determine the optimal harvest time, assess the number of days required to reach the optimal ripeness for consumption, and apply treatments to slow down the maturation process during export. Traditional methods are time-consuming and involve premature destruction of the fruits, leading to economic losses. Furthermore, these methods rely on a small sample size to estimate the conditions of the entire orchard.

In this article, we propose the design of a mini-quadcopter protected by a lightweight and resistant spherical casing and equipped with a robotic arm called "SAMBot: Spherical Aerial Manipulator Robot." This proposal addresses the identified problem by combining the protective advantages of the spherical casing in complex environments (such as tree

*Email address(es): icruzv@inaoep.mx

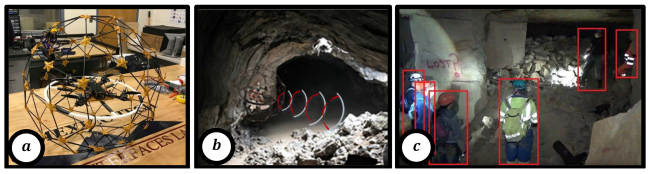


Figure 1: (a) Design for underground applications. (b) Obstacle detection with ultrasound. (c) Detection of workers in mines using the camera [11].

branches or leaves) with the manipulator's capability to access confined spaces. We present several challenges encountered during the simulation phase of SAMBot's design. Concurrently, we have been developing the physical design, considering challenges such as ensuring the combined weight of the sphere and manipulator does not exceed the drone's lifting capacity, and accounting for changes in the center of mass due to the manipulator's position, among other factors, to implement this design as a new entity in the Gazebo software while adhering to physical constraints.

Although we have not found in the literature a similar work, we now present some related works on spherical drones and aerial manipulators. A spherical drone (SD) is equipped with a spherical structure that provides protection and allows it to access confined and hazardous spaces without compromising the integrity and stability of the drone and its environment [10]. In [11], the SD is trained using deep learning techniques to fly in dangerous environments without GPS, such as mines for workers' rescue missions. Figure 1 shows some examples. Other tasks, such as autonomous underground navigation, survivor search, and infrastructure inspection, are considered in [12]. In [10] and [13], a drone is designed with a passive rotating spherical cage mechanism to stabilize it during collisions with structures damaged by earthquakes, fires, or explosions, such as buildings and bridges. In [14], this system employs cameras with computer vision and machine learning techniques to map public spaces and track city pedestrian flow. In [15], a cargo drone with a foldable modular spherical structure is used to lift packages weighing up to 500 grams. Authors in [16] present a more visually appealing application, where the drone casing creates three-dimensional images through the synchronized rotation and activation of multiple LED ribbons, arcing around the sphere to create virtual and augmented reality experiences, displaying 3D images for telepresence applications.

An Unmanned Aerial Manipulator (UAM) is a robotic system consisting of a drone equipped with a multi-DoF manipulator and various sensors and tools, such as claws or grippers. Its use expands the drone's ability to complete complex tasks and enhances the manipulator's maneuverability [17]. Since UAMs are the systems most similar to SAMBot, we have conducted an exhaustive search of the state-of-the-art UAMs [17–37]. According to these works, the most common applications for UAMs include trajectory tracking, position/orientation control, grasping and placement tasks, contact tasks, and force control (see Figure 2). The quadcopter is the most used UAV for UAMs, followed by the hexacopter and octocopter. 2-DoF manipulators are the most frequently used for equipping drones, as these systems are lighter and more compact in terms of weight and size. Furthermore, by leveraging the DoFs of the UAV itself, a more complex manipulator is not always necessary for certain tasks.

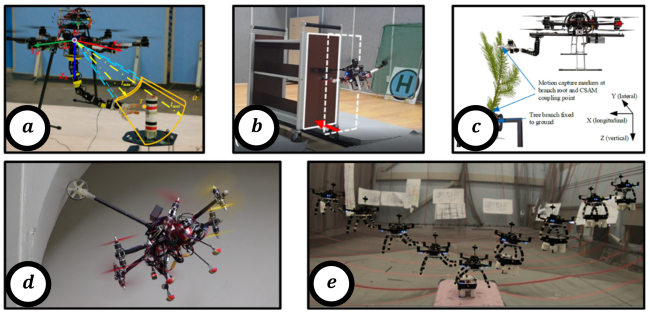


Figure 2: Featured applications of the UAM. (a) Grip/Placement. (b) Contact with quasi-static surface. (c) Coupling between the UAM and the object. (d) Contact with static surface. (e) Dynamic grip.

To obtain a mathematical description of UAMs, the Euler-Lagrange formulation is the most commonly used method for expressing the dynamic model, followed by the Newton-Euler formulation and the Recursive Newton-Euler formulation. Approximately 61% of the reviewed articles use a decoupled method, treating the manipulator as an additional disturbance. Additionally, we found that the control technique most commonly used for UAMs is PID control due to its straightforward design and implementation. Although the best results are not always achieved, most authors report that its performance is adequate for specific applications. Other techniques, such as Robust and Adaptive Control, are employed in more complex applications along with the use of a nonlinear dynamic model.

The rest of the paper is organized as follows: Section 2 provides a description of our proposed system and discusses the mathematical model for PID control actions, which represents an initial approach to controlling the drone in desired positions. Section 3 presents details of the design and results of the PID control. Finally, Section 4 summarizes the findings of this study and the future tendencies.

2 METHODOLOGY AND DESIGN

2.1 *Description of the System*

SAMBot consists of a drone protected by a spherical structure, equipped with a 3 DoF manipulator, capable of positioning and orienting its end-effector in the plane. The actuators in the design correspond to the MG90S servomotor.

http://www.imavs.org/

http://www.imavs.org/

The manipulator has three MPU6050 IMU sensors attached to each joint. Unlike an encoder, data acquisition with the IMU is simpler and does not require additional electronics nor direct mounting on the joint motor shaft, which makes it easier to install on a smaller and lighter structure that is useful in mobile applications such as drones. The end effector has an ESP32–Cam camera to estimate distances and implement computer vision techniques in future applications and a VL53L0X distance sensor near the camera lens to corroborate the measurements.

Since the payload for most mini-drones is very limited, all the components mentioned above were carefully selected to maintain the payload to the minimum while still having the necessary components to perform the tasks commonly presented in most PA scenarios. The spherical casing design, including the volume of the casing and the material selection, was as well iteratively designed to maintain the payload to the minimum. Moreover, the arm manipulator is incorporated to be part of the spherical casing to further reduce the payload.

According to [38], the dynamic equations of the quadcopter as a mechanical system are obtained using Newton's laws. Then the quadcopter motion is described by a nonlinear system of six differential equations. Also considering the mathematical equations of a planar revolute joint manipulator and supposing an n-link robot attached to the drone, the mathematical model of the manipulator is obtained using Euler-Lagrange formalism. After considering the manipulator connected to the quadcopter, values of the elements in the inertia tensor are changed due to the additional inertia of the manipulator links, the moment of inertia generated by each material point resulting in an inertia tensor of the combined system, also is determined the mutual influence between the quadcopter and the manipulator, considering the position displacement of the mass center of the combined system and the torque reaction forces $\tau_{(x,y,z)}$ acting on the quadcopter from the manipulator. The mathematical model of the quadcopter is described by the equations:

$$\begin{cases} m_m\ddot{x} = (s_\psi s_\phi + c_\psi s_\theta c_\phi)F + d_1 \\ m_m\ddot{y} = (-c_\psi s_\phi + s_\psi s_\theta c_\phi)F + d_2 \\ m_m\ddot{z} = -m_m g + (c_\theta c_\phi)F + d_3 \\ I_{xx}\ddot{\phi} = (I_{yy} - I_{zz})\dot{\theta}\dot{\psi} - I_{xz}(\dot{\phi}\dot{\theta} + \ddot{\psi}) + \tau_x + d_4 \\ I_{yy}\ddot{\theta} = (I_{zz} - I_{xx})\dot{\phi}\dot{\psi} + I_{xz}(\dot{\phi}^2 - \dot{\psi}^2) + \tau_y + d_5 \\ I_{zz}\ddot{\psi} = (I_{xx} - I_{yy})\dot{\phi}\dot{\theta} + I_{xz}(\dot{\theta}\dot{\psi} + \ddot{\phi}) + \tau_z + + d_6 \\ \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) + \mathbf{d}_7 = \boldsymbol{\tau} \end{cases} \quad (1)$$

where s_x and c_x denotes $\sin(x)$ and $\cos(x)$, respectively; m_m is the overall mass of the whole system; $\ddot{x}, \ddot{y}, \ddot{z}$ are the linear accelerations of the quadcopter with respect to the inertial reference frame; I_{xx}, I_{yy}, I_{zz} represents the inertia tensor of the combined system; ψ, θ, ϕ are the $Yaw - Pitch - Roll$ angles that describe the drone orientation, angular velocity $(\dot{\psi}, \dot{\theta}, \dot{\phi})$ and angular acceleration $(\ddot{\psi}, \ddot{\theta}, \ddot{\phi})$; F is the lift of the airframe; τ_x, τ_y, τ_z are the torques of the three axes of the airframe, respectively; I_{xz} represents the inertia product of the combined system (since the manipulator operates on $x - z$ plane); g is the gravitational acceleration constant. For the manipulator: $\mathbf{M}(\mathbf{q})$ is the inertia matrix; $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the coriolis and centripetal forces matrix; $\mathbf{g}(\mathbf{q})$ is the vector accounting for gravitational forces; $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$ are the vector of generalized coordinates, velocities and accelerations of the manipulator joints, respectively; and $\boldsymbol{\tau}$ are the actuators' torques. Finally $d_i (i = 1, ..., 6)$ represents the external disturbance in each channel, and $\mathbf{d}_7 = [d_{71}, ..., d_{7n}]$ indicates the external interference of the n-link of the manipulator, which the quadcopter and external environment may cause.

2.2 Experimental Simulation Tests

The SAMBot was designed in SolidWorks, and the URDF (Unified Robot Description Format) file was exported using the sw2urdfSetup.exe plugin. The URDF has subsequently been modified to a .urdf.xml format for incorporation with the Parrot AR Drone 2.0 in the sjtu-drone-quadrator-simulation in Gazebo and communication using ROS2 Humble, installed in Ubuntu 22.04. During this stage, some modifications were realized to join the system with the proper dimensions, mass parameters, and inertia tensor of the links, the joint type, and the action limits. In this sense, we have ensured a faithful representation of the proper conditions to evaluate the drone's performance in the planned tasks.

To obtain these physical parameters, the mass of each element was first estimated in Ultimaker-Cura-5.2.1, using the PLA material with a density of 25%; the estimated mass was used to calculate the physical parameters with the "Physics properties" tool from SolidWorks.

The joint-trajectory-controller of the ROS2 control package has been implemented to control the manipulator. In turn, a PID controller has been implemented to control the drone, manually adjusting the gains online using a graphical user interface (GUI) developed in this work. In the experimental tests, the ability of SAMBot to remain hovering while extending and retracting the manipulator was analyzed, and basic movements in x, y, z, and yaw were performed. The results are shown in the following section.

3 RESULTS AND DISCUSSIONS.

The design of SAMBot is shown in Figures 3 and 4. Its design is symmetrical and takes advantage of the spherical structure so that the manipulator is part of it.

The previous section mentioned the sensors and actuators that will be used when applied in a real environment. The dimensions were checked carefully by using precision instruments, and the most relevant parts were printed for assembly tests. These elements are shown in Figure 5.

Our design was coupled with the Parrot AR Drone 2.0 in Gazebo to perform the simulations, as shown in Figure 6.

The GUIs were used to send control commands to the Drone and manipulator. The most relevant part is highlighted in green in Figure 7; from there, the PID gains were adjusted online to stabilize the Drone.

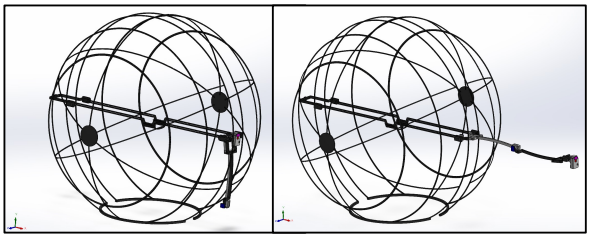


Figure 3: SAMBot design. Manipulator folded (left) and deployed (right).

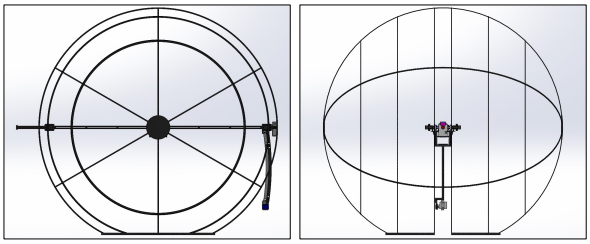


Figure 4: Side (left) and front (right) view of SAMBot with folded manipulator.

Figure 8 shows the error signals for x, y, z, and yaw when sending a command sequence to the Drone. The blue line is the current error, and the dashed red line indicates the zero error reference for an ideal case. The sequence is described below following the numbering marked in Figure 8:

1. A take-off signal was sent to the Drone to reach a reference of 2 m along the z-axis while the manipulator was retracted to its initial position (as seen in Figure 6). This action affected the x and z axes. In both cases (mainly for x), it was more difficult to reach the reference; however, the system did not present oscillations that affected its stability.

2. The manipulator was ordered to be fully deployed while the Drone remained hovering. Since the PID parameters were primarily tuned for this case, the error is nearly zero in all plots.

3. When the manipulator was folded back home, a behavior similar to that of point (1) was observed.

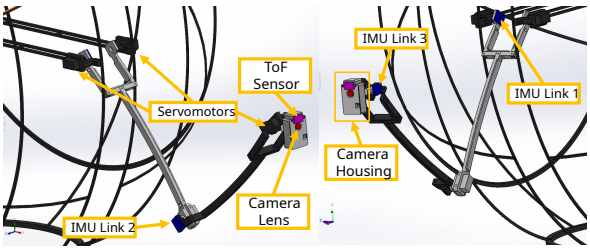


Figure 5: Assembly of sensors and actuators..

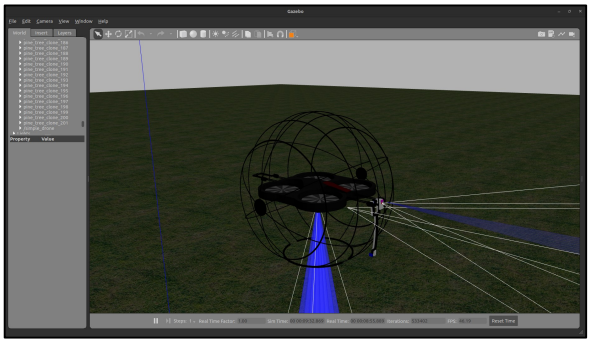


Figure 6: SAMBot simulation environment in Gazebo.

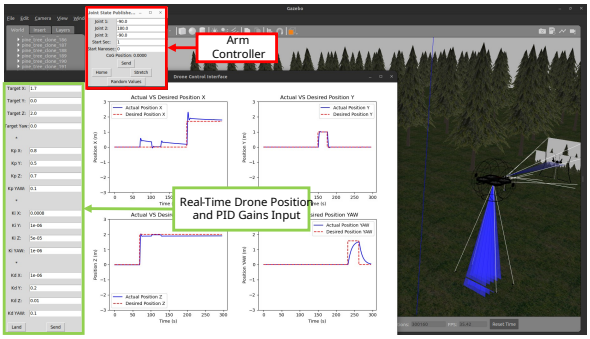


Figure 7: SAMBot simulation environment in Gazebo with GUI's for tuning and control.

4. It moved one meter in the positive direction of the y-axis. The error graph shows that the error decreases rapidly without considerable oscillations.

5. The same movement as the previous point, but this time in a negative sense.

6. Positive motion in yaw of 1.57 rad.

7. Negative movement in yaw of -1.57 rad relative to its current position. The graph shows that the yaw error tends practically to zero in stationary state.

8. Finally, the Drone is moved 1.3 meters towards the front (on the x-axis), and it was observed to have small oscillations until reaching a stationary error.

A similar sequence is illustrated in Figure 9, this time comparing the system's desired position with its actual position. The graphs show that the signals on the x and z axes are similar when the manipulator is fully folded, as shown in (a), compared to when it is fully deployed, as shown in (b). Some screenshots of the system are shown in Figure 10 while the simulation was running. A video of the simulation described above can be found at `https://www.youtube.com/watch?v=LFM5YPrHYyM`.

http://www.imavs.org/

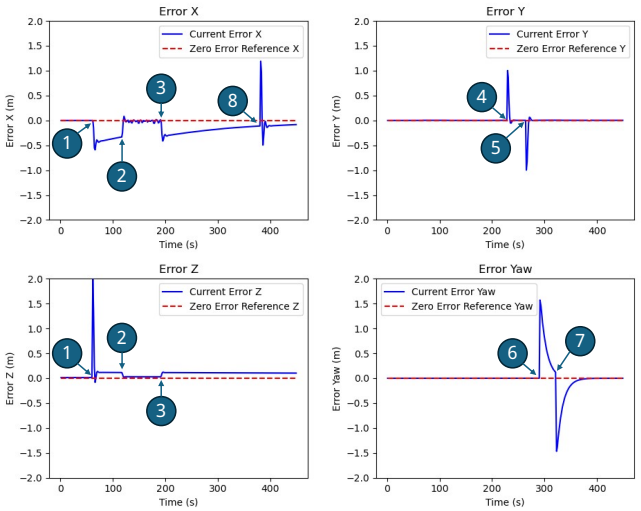


Figure 8: Error signals of x, y, z, yaw during a sequence of commands sent to the Drone.

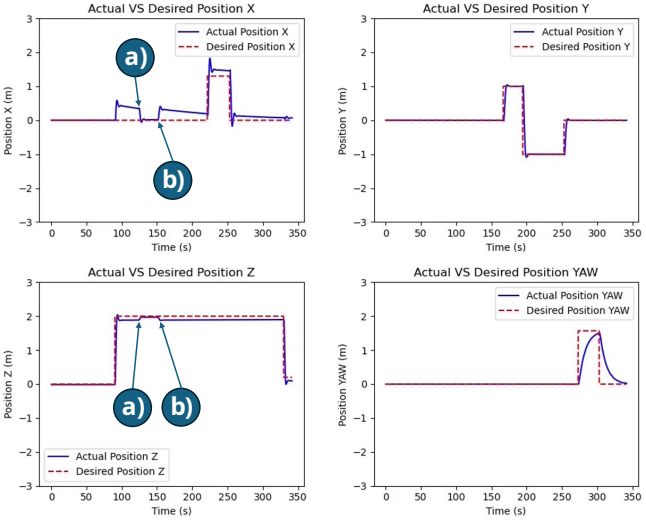


Figure 9: Desired position vs actual position signals.

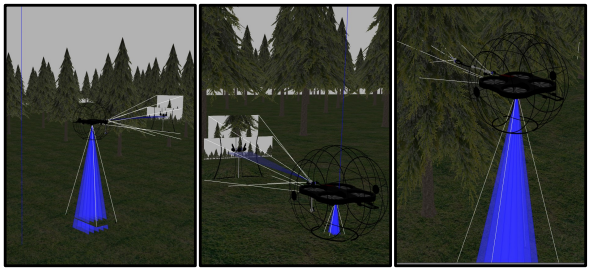


Figure 10: SAMBot simulation images.

4 CONCLUSION

The design of "SAMBot: Spherical Aerial Manipulator Robot" has been presented, a novel system consisting of a mini-drone protected by a spherical structure and equipped with a manipulator. The protection provided by the sphere and the ability of the manipulator to access confined spaces allows our robot to expand to various applications, such as accessing complex forest environments to obtain information on crops, search and rescue in mines or collapsed buildings, infrastructure inspection, autonomous underground navigation, among others. In our extensive research, we found a surprising gap in the literature-no studies on similar robots. This stark absence underlines the pressing need for research on applications of spherical drones with manipulation capabilities. As a contribution to this field, we have incorporated our design into Gazebo in ROS2 Humble, and it will be freely available to researchers who wish to delve into this unexplored area of research. A PID controller has been implemented to stabilize the Drone. We have developed a graphical user interface (GUI) that allows users to manually adjust the gains online. In future work, we will implement more complex control strategies, such as Robust or Adaptive Control combined with Artificial Intelligence strategies, to compensate for the manipulator's dynamic movement and the moment of inertia generated by our complete system.

REFERENCES

[1] Mostafa Hassanalian and Abdessattar Abdelkefi. Classifications, applications, and design challenges of drones: A review. *Progress in Aerospace sciences*, 91:99–131, 2017.

[2] J Daniel Olivares-Figueroa, Israel Cruz-Vega, JM Ramírez-Cortés, P Gómez-Gil, and J Martínez-Carranza. A compact approach for emotional assessment of drone pilots using bci. In *12th international micro air vehicle conference, Puebla, México*, pages 57–62, 2021.

[3] S Ahirwar, R Swarnkar, S Bhukya, and G Namwade. Application of drone in agriculture. *International Journal of Current Microbiology and Applied Sciences*, 8(01):2500–2505, 2019.

[4] Gopal Dutta and Purba Goswami. Application of drone in agriculture: A review. *International Journal of Chemical Studies*, 8(5):181–187, 2020.

[5] Mohammad Fatin Fatihur Rahman, Shurui Fan, Yan Zhang, and Lei Chen. A comparative study on application of unmanned aerial vehicle systems in agriculture. *Agriculture*, 11(22), 2021.

[6] João Valente, Rodrigo Almeida, and Lammert Kooistra. A comprehensive study of the potential application of flying ethylene-sensitive sensors for ripeness detection in apple orchards. *Sensors*, 19(2):372, 2019.

http://www.imavs.org/

[7] Salua Hamaza, Ioannis Georgilas, Manuel Fernandez, Pedro Sanchez, Thomas Richardson, Guillermo Heredia, and Anibal Ollero. Sensor installation and retrieval operations using an unmanned aerial manipulator. *IEEE Robotics and Automation Letters*, 4(3):2793–2800, 2019.

[8] André Farinha, Raphael Zufferey, Peter Zheng, Sophie F. Armanini, and Mirko Kovac. Unmanned aerial sensor placement for cluttered environments. *IEEE Robotics and Automation Letters*, 5(4):6623–6630, 2020.

[9] Kenjiro Tadakuma, Carl John Salaan, Eri Takane, Yoshito Okada, Kazunori Ohno, and Satoshi Tadokoro. Design of aerial manipulator suitable for a uav with two passive rotating hemispherical shells. In *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–6, 2018.

[10] Adrien Briod, Przemyslaw Kornatowski, Jean-Christophe Zufferey, and Dario Floreano. A collision-resilient flying robot. *Journal of Field Robotics*, 31(4):496–509, 2014.

[11] Javad Shahmoradi, Amir Mirzaeinia, Pedram Roghanchi, and Mostafa Hassanalian. Monitoring of inaccessible areas in gps-denied underground mines using a fully autonomous encased safety inspection drone. In *AIAA Scitech 2020 Forum*, page 1961, 2020.

[12] Paolo De Petris, Huan Nguyen, Mihir Dharmadhikari, Mihir Kulkarni, Nikhil Khedekar, Frank Mascarich, and Kostas Alexis. Rmf-owl: A collision-tolerant flying robot for autonomous subterranean exploration. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 536–543. IEEE, 2022.

[13] Shoma Mizutani, Yoshito Okada, Carl John Salaan, Takuma Ishii, Kazunori Ohno, and Satoshi Tadokoro. Proposal and experimental validation of a design strategy for a uav with a passive rotating spherical shell. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1271–1278, 2015.

[14] Aldo Sollazzo, Armando Trento, and Efilena Baseta. Machinic agency - implementing aerial robotics and machine learning to map public space. *35th eCAADe Conference - Sharing Computational Knowledge*, 2:611–618, 2017.

[15] P.M. Kornatowski, S. Mintchev, and D. Floreano. An origami-inspired cargo drone. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6855–6862, 2017.

[16] Wataru Yamada, Kazuhiro Yamada, Hiroyuki Manabe, and Daizo Ikeda. isphere: self-luminous spherical drone display. In *Proceedings of the 30th annual ACM symposium on user interface software and technology*, pages 635–643, 2017.

[17] Xu Wei-hong, Cao Li-jia, and Zhong Chun-lai. Review of aerial manipulator and its control. *Int. J. Robot. Control Syst*, 1(3):308–325, 2021.

[18] Xiangdong Meng, Yuqing He, Qi Li, Feng Gu, Liying Yang, Tengfei Yan, and Jianda Han. Contact force control of an aerial manipulator in pressing an emergency switch process. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2107–2113. IEEE, 2018.

[19] Guangyu Zhang, Yuqing He, Bo Dai, Feng Gu, Jianda Han, and Guangjun Liu. Robust control of an aerial manipulator based on a variable inertia parameters model. *IEEE Transactions on Industrial Electronics*, 67(11):9515–9525, 2019.

[20] Yen-Chen Liu and Chi-Yu Huang. Ddpg-based adaptive robust tracking control for aerial manipulators with decoupling approach. *IEEE Transactions on Cybernetics*, 52(8):8258–8271, 2021.

[21] Weixuan Zhang, Maximilian Brunner, Lionel Ott, Mina Kamel, Roland Siegwart, and Juan Nieto. Learning dynamics for improving control of overactuated flying systems. *IEEE Robotics and Automation Letters*, 5(4):5283–5290, 2020.

[22] Karen Bodie, Maximilian Brunner, Michael Pantic, Stefan Walser, Patrick Pfändler, Ueli Angst, Roland Siegwart, and Juan Nieto. Active interaction force control for contact-based inspection with a fully actuated aerial vehicle. *IEEE Transactions on Robotics*, 37(3):709–722, 2020.

[23] Kyunam Kim, Patrick Spieler, Elena-Sorina Lupu, Alireza Ramezani, and Soon-Jo Chung. A bipedal walking robot that can fly, slackline, and skateboard. *Science Robotics*, 6(59):eabf8136, 2021.

[24] Dimitris Chaikalis, Farshad Khorrami, and Anthony Tzes. Adaptive control approaches for an unmanned aerial manipulation system. In *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 498–503. IEEE, 2020.

[25] Yaqi Wu, Jin Song, Jiabi Sun, Fenfang Zhu, and Haoyao Chen. Aerial grasping based on vr perception and haptic control. In *2018 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 556–562. IEEE, 2018.

http://www.imavs.org/

[26] Yanjie Chen, Weiwei Zhan, Bingwei He, Lixiong Lin, Zhiqiang Miao, Xiaofang Yuan, and Yaonan Wang. Robust control for unmanned aerial manipulator under disturbances. *Ieee Access*, 8:129869–129877, 2020.

[27] Joshua Fishman, Samuel Ubellacker, Nathan Hughes, and Luca Carlone. Dynamic grasping with a" soft" drone: From theory to practice. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4214–4221. IEEE, 2021.

[28] Dimos Tzoumanikas, Felix Graule, Qingyue Yan, Dhruv Shah, Marija Popovic, and Stefan Leutenegger. Aerial manipulation using hybrid force and position nmpc applied to aerial writing. *arXiv preprint arXiv:2006.02116*, 2020.

[29] Z Samadikhoshkho, S Ghorbani, F Janabi-Sharifi, and K Zareinia. Nonlinear control of aerial manipulation systems. *Aerospace Science and Technology*, 104:105945, 2020.

[30] Karen Bodie, Maximilian Brunner, Michael Pantic, Stefan Walser, Patrick Pfändler, Ueli Angst, Roland Siegwart, and Juan Nieto. An omnidirectional aerial manipulation platform for contact-based inspection. *arXiv preprint arXiv:1905.03502*, 2019.

[31] Emre Yilmaz, Hammad Zaki, and Mustafa Unel. Nonlinear adaptive control of an aerial manipulation system. In *2019 18th European control conference (ECC)*, pages 3916–3921. IEEE, 2019.

[32] Chioniso Kuchwa-Dube and Jimoh O Pedro. Quadrotor-based aerial manipulator altitude and attitude tracking using adaptive super-twisting sliding mode control. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 144–151. IEEE, 2019.

[33] James R Kutia, Karl A Stol, and Weiliang Xu. Aerial manipulator interactions with trees for canopy sampling. *IEEE/ASME transactions on mechatronics*, 23(4):1740–1749, 2018.

[34] Vinh Nguyen, Anton Saveliev, and Andrey Ronzhin. Mathematical modelling of control and simultaneous stabilization of 3-dof aerial manipulation system. In *International Conference on Interactive Collaborative Robotics*, pages 253–264. Springer, 2020.

[35] Chang Zhiyuan, Luo Yanyang, Shao Yanhua, Chu Hongyu, Wu Bin, Huang Mingqi, and Yunbo Rao. Fuzzy sliding mode control for rotorcraft aerial manipulator with extended state observer. In *2020 Chinese Automation Congress (CAC)*, pages 1710–1714. IEEE, 2020.

[36] Dongjae Lee, Hoseong Seo, Inkyu Jang, Seung Jae Lee, and H Jin Kim. Aerial manipulator pushing a movable structure using a dob-based robust controller. *IEEE Robotics and Automation Letters*, 6(2):723–730, 2020.

[37] Guangyu Zhang, Yuqing He, Bo Dai, Feng Gu, Liying Yang, Jianda Han, and Guangjun Liu. Aerial grasping of an object in the strong wind: Robust control of an aerial manipulator. *Applied Sciences*, 9(11):2230, 2019.

[38] Xitong Guo, Pingjuan Niu, Di Zhao, Xia Li, Shan Wang, and An Chang. Model-free controls of manipulator quadrotor uav under grasping operation and environmental disturbance. *International Journal of Control, Automation, and Systems*, 20(11):3689–3705, 2022.

http://www.imavs.org/

Dexterity Assessment of Fully-Actuated UAVs

Chantelle Singh, Shahab Kazemi, Joao Buzzato, and Karl Stol
Department of Mechanical and Mechatronics, University of Auckland, New Zealand.
Corresponding author Email: shahab.kazemi@auckland.ac.nz

http://www.imavs.org/

ABSTRACT

Aerial manipulation uses unmanned aerial vehicles (UAVs) to perform interaction tasks with the environment. Many aerial applications are conducted safer and more efficient when using UAVs to replace human workers. An essential aspect of aerial manipulation is dexterity, which is generally used to describe the ability of human or robotic hands to perform tasks such as moving and rotating an object. Currently, there is limited research on the dexterity of fully-actuated UAVs. Furthermore, although there are some papers on standardized testing for aerial manipulation, there still needs to be a universally accepted method to measure dexterity or aerial manipulation capabilities in fully-actuated UAVs. This work explores the concept of dexterity in the context of fully-actuated UAVs without complex manipulator degrees of freedom (DoF) and investigates how UAV dexterity can be quantified. In this paper, three standardised UAV dexterity tests are developed and experimented on a tilted-rotor octocopter UAV to demonstrate how the feasible dexterity tests would be conducted in a controlled environment.

1 INTRODUCTION

Aerial manipulation includes a manipulator attached to a UAV hovering in the air interacting with the environment. Recently, there has been increased interest in aerial manipulation, frequently prescribed for high-altitude tasks that could be more efficient or safer than using human workers. Examples of aerial manipulation applications include canopy sampling, natural disaster clean up, high-rise window cleaning, rescue missions, and maintenance tasks [1, 2, 3].

A vital characteristic needed in manipulation is *dexterity* as described in several research studies [4, 5, 6, 7]. However, dexterity is a concept typically applied to humans and robotic hands/arms. The literature describes it as a measure of a human hand's ability to interact with the environment involving using the fingers skillfully or doing interaction tasks such as precise hand movements while manipulating objects. Despite its significance, dexterity remains a relatively unexplored characteristic in the research area of UAVs. Furthermore, more research is needed to assess the dexterity of UAVs without the added complexities introduced by multi-DoF arms.

To date, there has not yet been a standardized dexterity assessment that thoroughly examines the capabilities of fully-actuated UAVs, which have independent control of all six DoF. This study aims to quantify the dexterity of such UAVs without additional DoFs via a manipulator arm and to identify variations in dexterity across different UAVs through experimental tests. Considering the advantages and additional capabilities associated with full actuation is essential to this research. By doing so, we can compare the dexterity of different fully-actuated UAVs and identify areas where dexterity may be lacking.

Several studies have investigated human dexterity resulting in various dexterity tests available for the hand [8]. A common human dexterity test is the nine peg-in-hole test which assesses finger dexterity [9]. This test involves the human participant placing pegs in holes with a time constraint. These dexterity tests are designed to diagnose neurodegenerative diseases such as Parkinson's or multiple sclerosis, which affect a person's dexterity.

Dexterity can also be applied in the field of robotics. Researchers aim to enhance robotic hand manipulation capabilities, often comparing them to human abilities. Standardized dexterity tests have been developed that are influenced by human dexterity research to evaluate robotic arm's performance [5]. Elangovan et al. developed nineteen tests covering object manipulation and tool interaction to find an overall dexterity score. Metrics like reachable workspace and force closure are standard in dexterity research, defining the space a manipulator can access and its ability to withstand forces and torques [6, 4, 10].

An example of a standardised UAV test is seen in the study by Suarez et al. [11]. Several aspects of a UAV and manipulator arm combination were tested, such as accuracy and manipulation capability. One test involved checking that the UAV could apply force on a wall for 5 s which is similar to the force closure concept discussed previously. However, this research has several limitations, such as not focusing on dexterity and being limited to under-actuated UAV analysis.

Despite the limited availability of dexterity tests designed explicitly for UAVs and even fewer tests that do not depend on a manipulator arm, various research studies have conducted experimental tests to validate the design and control architectures of these UAVs. These tests could serve as suitable dexterity measures because they evaluate the UAV's ability to perform precise, controlled, and coordinated movements that

http://www.imavs.org/

are helpful for manipulation. A typical experiment frequently seen in several studies is position holding, which assesses the UAV's ability to accurately maintain a specific position, reflecting its maneuverability and control precision. For example, a study by Jiang et al. [12] developed a fully-actuated hexrotor with optimized fixed tilt-rotor angles designed for aerial manipulation. The UAV experimental tests in this study show similarities to the workspace analysis in dexterity research. The test involved hovering on one spot at a roll angle with zero translation, which measures how far the UAV can reach. Kamel et al. [13] developed experimental tests involving the hexacopter rotating around one axis, which showed how the UAV was stable at different orientations. This maneuver is impossible for a typical under-actuated UAV.

Several experiments analyzed precision, which is vital for manipulation. Orsag et al. [7] examined an under-actuated quadcopter conducting insertion task using a manipulator arm. A success rate was measured in performing these tasks. However, a limitation of this research is that this test does not directly assess the dexterity of a UAV.

The concept of force closure is not usually considered for UAVs; however, it can be considered when aerial manipulation is involved. With sufficient force applied by the UAV, manipulation would be more precise because it could maintain contact and complete actions, resulting in improved dexterous manipulation. A study also conducted by Jiang et al. [14] analysed force closure and developed their UAV to instantaneously handle any wrench (combination of force and torque). In order to analyse forces and torques in each axis of the hexrotor UAV, a force/torque ellipsoid was created for rotor angles of 0º, 10º, 20º and 30º.

This paper is organised as follows. Section 2 explores what dexterity means in the context of UAVs and the requirements of a standardised UAV dexterity test. Section 3 describes the physical setup for the case study experiments conducted on a fully-actuated octocopter. This section also describes the relationships between the systems and components in this research study. Section 4 introduces the design of the dexterity tests and the experimental approach for each test. The paper then discusses the results from the case study experiments for each test in Section 5.

2 Defining Terms and Establishing a Criteria for Effective Testing

2.1 UAV Dexterity Definition

A definition of UAV dexterity can be synthesized by drawing from existing definitions for human dexterity, robotic hands, and dexterous biological aerial animals, such as hummingbirds. Hummingbirds display exceptional flight dexterity through precise hovering stability, accurate control of pitch, roll, and yaw, and the ability to dynamically adjust their body orientation. They can independently maneuver each wing, enabling sharp turns and agile navigation. This fine control is crucial for making precise contact with flowers

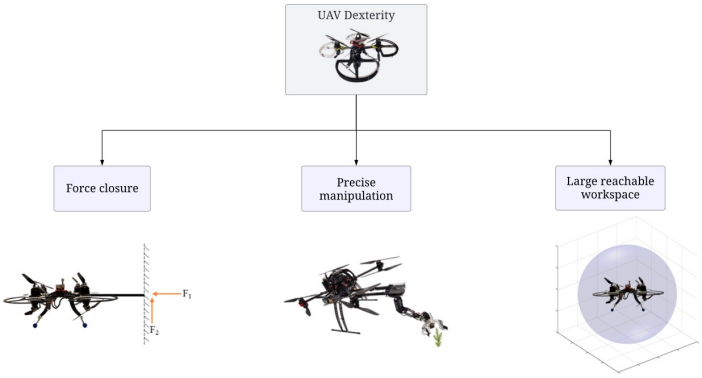


Figure 1: Conceptual framework of UAV dexterity

using their beaks, demonstrating advanced aerial manipulation. As discussed in Section 1, dexterity is a contributing factor in object manipulation skills. For UAVs designed for aerial manipulation, dexterity directly impacts proficiency in precisely handling objects during interactions.

A fundamental aspect of dexterity is force closure [10], which in the context of UAVs denotes the ability to resist external forces and torques during manipulation tasks, ensuring stable and sustained contact without slippage or loss of control over the manipulated object. Furthermore, reachability is a common consideration in robotic arm dexterity tests, as described in Section 1. The reachability of a UAV, independent of fixed manipulator arms, is helpful for its dexterity. The reachable locations are the UAV's workspace while hovering. That is, the maximum roll, pitch and yaw.

Several metrics similar to dexterity exist in the literature, such as manoeuvrability and agility [15, 16]. While these metrics contribute to UAV performance, they do not inherently encapsulate the precision and accuracy intrinsic to dexterity. Moreover, drawing inspiration from biological models, such as hummingbirds, provides valuable insights into defining UAV dexterity. Hummingbirds are known as one of the most skilful aerial manipulators in nature, as described by Haque et al. [17]. Figure 1 summarizes the definition of UAV dexterity developed in this research. This definition is utilized to create tests for measuring UAV dexterity.

2.2 Dexterity Test Requirements

To ensure compliance with existing standardized testing protocols and adequately address the unique characteristics of fully-actuated UAVs, criteria for a standardized dexterity assessment must be established. These criteria are used to develop each test, and many feasible requirements will be incorporated into the tests. According to several existing standardised testing schemes [5, 11], common characteristics involve repeatability, generalizability, affordability, and accessibility. Additionally, the dexterity tests should include being quantifiable, having a transferable score to application capability,

and being safe. Finally, the measure should be valid and reliable. Furthermore, since this research is focused on UAV dexterity, the tests also require to be independent of external attachments such as manipulator arms and therefore must not use an attachment with more that one DoF. Another aspect of this research is the test distinguishes dexterity across fully-actuated UAVs.

3 PHYSICAL SETUP AND UAV DESCRIPTION

3.1 Experimental Methods and Tools

Each experiment in this research uses several experimental tools, which are discussed in this Section. Figure 2a depicts an overview of the experimental setup. An offboard interface using MATLAB allows sending translational or rotational setpoints to the UAV. The Vicon tracker software is used as the motion capture system enabling precise tracking of the UAV's movements in 3D space. Additionally, the Vicon tracker enables relative displacements between different objects to be tracked.

The UAV chosen as a test subject is a fully-actuated octocopter (refer to Figure 4) with a co-axial stacked configuration featuring four rotors mounted on both the top and bottom levels. Each rotor is fixed at a 31 º angle, enabling a generalized wrench through the motor map. This UAV model was previously developed in a research study by Mendes Souza et. al. [18].

3.2 Relationships Between Systems

The interconnection among various systems within the experimental setup is depicted in Figure 2b. The offboard control system transmits setpoints via MAVlink User Datagram Protocol (UDP) packets, which the UAV receives through the Wi-Fi module. Subsequently, the PX4 flight controller utilizes these setpoints to command the UAV's movements during the dexterity test. Simultaneously, information regarding the UAV's spatial location, acquired through the reflective markers tracked by the motion capture cameras positioned around the laboratory, is sent back to the offboard control PC. This feedback loop enables the calculation of the distance required to reach the setpoint, allowing for adjustments to the UAV's movements until the setpoint is reached. Following each test, the flight log is downloaded for data analysis. Moreover, experimental setup in the motion capture laboratory is illustrated in Figure 2a. A PC, connected to a webcam positioned within the experimental area, also monitors the ongoing experiment. The experiment area is an enclosed netted space where the UAV tests occur.

4 DEXTERITY TESTS DESIGN

Each subsection below begins with an exploration of the rationale behind conducting the test and its significance within the context of UAV dexterity assessment. Following this, a detailed discussion of each test's design takes place. Then, a quantitative scoring metric formulation, which facilitates the standardized evaluation of UAV dexterity, is established. Furthermore, each subsection concludes with an experimental case study conducted on the fully-actuated octocopter.

4.1 Test I: Reachability

4.1.1 Test Design

The test design process draws inspiration from existing workspace analysis research of robotic arms to devise a modified test. The test looks at determining the reachable workspace of a UAV during hovering while also considering the test requirements outlined in Section 2.2. UAVs can move freely in 3D space, resulting in an effectively infinite workspace for translation. Thus, to create a test applicable to UAVs in general, the test is adapted to analyse the space reachable via rolling and pitching. The test is devised by evaluating the maximum roll and pitch angles the UAV could attain before reaching 20% away from the saturation limits. Since all UAVs can have maximum yaw (360º), yaw limits are not analysed.

This test is designed to comprehensively assess the reachable workspace of any fully-actuated UAV. It is adaptable for use on various UAV configurations, including octocopters, hexacopters or dodecacopters, which possess independent control over all six DoFs. Examining a UAV's reachable limits obtains a reachable workspace area. These limits involve determining the maximum roll and pitch angles for each combination of roll and pitch. Only a quarter of these points are determined through experiments as the UAV is symmetrical about each axis. Therefore, the negative limits are a mirror image of the positive limits.

Several data points are recorded at each limit to ensure a robust measure of the UAV's reachability workspace. These data points formed a cloud, from which a line of best fit is derived. Subsequently, an area of the first quarter of the UAV's reachability workspace is analysed to determine the score.

The scoring system is based on the area derived from the enclosure of the border of plotted roll and pitch limits calculated into a percentage as shown in Equation 1. This is to normalise the score and make it easier to compare with other UAVs.

$$R = \frac{A}{360^2} \cdot 100\% \tag{1}$$

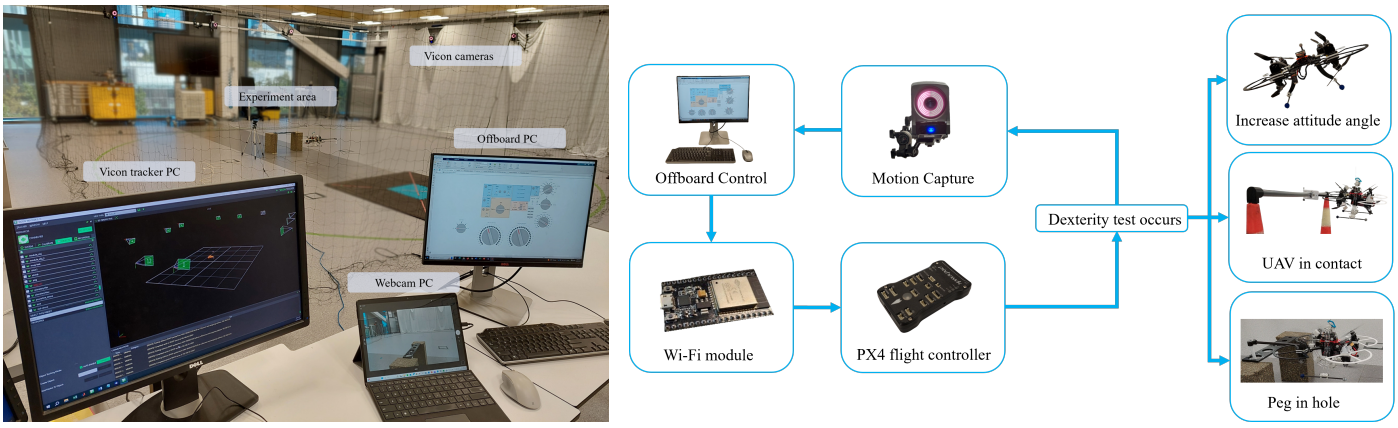
where R is the reachability test score and A is the reachable workspace area.

The maximum score achievable is: $\frac{360^2}{360^2} \cdot 100\% = 100\%$. The score results from the fact that the maximum roll and pitch angles possible is ±180º. Therefore, the side lengths of the area graph could only range from +180 to -180 in the roll and pitch axes.

4.1.2 Experiments

Considering time constraints during experiments and the assumption that the UAV is balanced (with the CoM in the cen-

http://www.imavs.org/



(a) Physical setup

(b) Connection between different elements used in experiments

Figure 2: Experimental setup

http://www.imavs.org/

tre of the UAV), the roll and pitch angle limits are expected to be symmetrical at about zero. Hence, only the positive roll and pitch limits are tested, and a quarter of the reachable plot is found through experiments. Multiple tests are conducted to identify each outer limit.

The experimental procedure for this test involves using an increasing ramp signal of pitch for a constant roll angle, up to the maximum roll angle with zero pitch. The maximum pitch angle the UAV can achieve with the particular constant roll angle is reached when it utilises 80% of its thrust capability, indicating it is within 20% of its motor PWM limits.

The pitch signal rate is slow enough to allow time for the PWM signal to update on the offboard PC to be monitored by a human, ensuring that saturation is unlikely to occur. The process of monitoring PWM signals during offboard control is depicted in Figure 3 where the pitch angle at PWM levels 20% away from saturation is used as the limit. An experiment testing a high roll angle is illustrated in Figure 4a, while a test where the UAV maintains a simultaneous non-zero roll and pitch angle is depicted in Figure 4b.

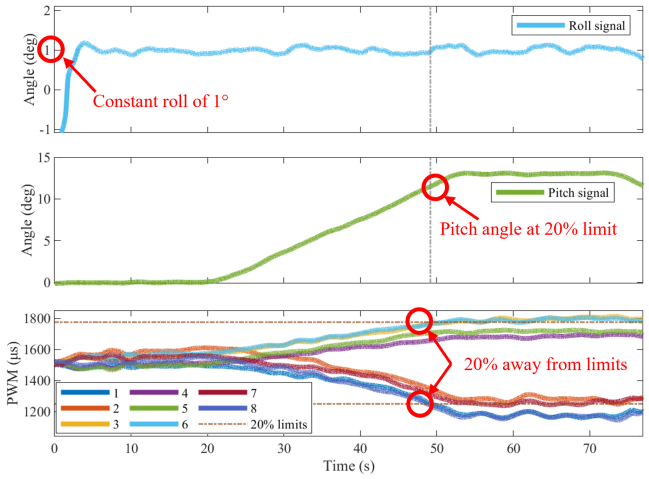
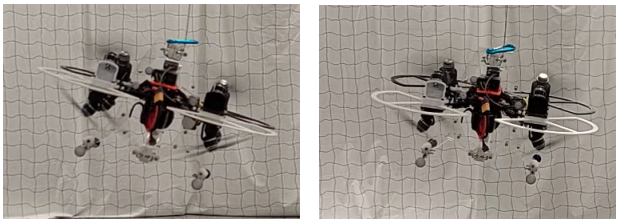


Figure 3: Finding the pitch limit for 1° roll

4.2 *Test II: Contact Testing*

4.2.1 Test Design

Several aerial manipulation applications like welding, powerline testing, and inspection using pushing mechanisms are examples of where force closure is important to consider [19, 20, 21]. This is due to the necessity of sustained contact while applying a force. The proposed test inspired by the concept of force closure discussed in Section 1, is to use a rigidly attached carbon fiber rod with an end-effector comprising a spring. The set-up and tools described in Section 3 allow for the translation of the UAV towards a flexible rod, establishing contact, and continuing to compress the spring until the



(a) High roll angle

(b) Roll and pitch angle combination

Figure 4: Experiments

http://www.imavs.org/

PWM signals are within 30 % of their limits. The 30 % from the limits is implemented as a safety factor to avoid the UAV flipping over the contact rod if it gets too close to saturating. This choice balances demonstrating sustained force with minimizing the risk of damage posed by physical objects in the UAV experiment area. Additionally, the UAV must compress for 30 s to demonstrate stability and not instantaneously capable of applying this force. Several research studies into UAV contact testing opted to maintain contact for 10 s [21, 22]. Hence, this research chose the required contact testing time to be 30 s, emphasizing sustained contact. The spring will compress, and using motion capture technology, the displacement is measured. The compression force can be calculated with the displacement and the known spring stiffness. This experiment is repeated for 0 °, 45 °, 90 °, -45 °, and -90 ° angles. These angles are selected based on previous research on contact-based inspection UAVs [20].

The scoring system measures the force closure ability of UAVs by analysing their capacity to maintain sustained contact at different angles. The specific angles are based on their relevance to contact testing related applications, as in previous studies [20]. This test evaluates the UAV's ability to apply a force that maintains PWM signals within 30 % saturation for a sustained period, thereby quantifying the force the UAV can exert over time. Two conditions are applied to the test to analyse these characteristics. The first condition is that the compression period should exceed or equal to 30 s. Additionally, the maximum deviation in attitude angle must be below 10 °. The scoring equation is described by Equation 2.

$$C = \frac{\sum_1^5 F_n.P_n}{W} \tag{2}$$

where C is the contact test score, F_n is the force exerted by UAV at contact angle *n,* W is the UAV weight, P_n is 0 or 1 depending on whether contact testing dexterity test conditions have been met, and n is equal to 1, 2, 3, 4, or 5 (which are associated with angles 0º, 45º, 90º, -45º, -90º).

4.2.2 Experiments

The experimental setup involved the use of an acrylic interaction frame comprised five 3D-printed blocks positioned at various angles, serving to support a carbon fiber rod. Additionally, a 3D-printed end-effector was attached to the rod's end, featuring a linear spring and a sliding mechanism designed to compress the spring upon contact. The displacement of the spring could be tracked using a Vicon tracker, providing data on the compression force applied by the UAV.

The experimental procedure begins with the UAV is manually armed using a transmitter and switched to offboard control mode, where the UAV autonomously reaches the predefined set points corresponding to the rod, as determined through motion capture markers. To demonstrate sustained force within the UAV's saturation limits, a force is applied by the UAV sufficient to bring the PWM signals within 30 % saturation for a sustained period, approximately ranging between 2 and 3 N for the case study octocopter. Finally, flight log and motion capture recording data were collected to calculate the displacement of the spring on the end-effector and the coaxial compression force. Figure 5 illustrates the -45º experiment.

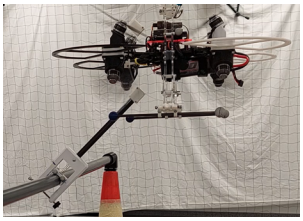


Figure 5: Experiment for -45º contact angle

4.3 Test III: Peg-in-Hole

4.3.1 Test Design

The peg-in-hole test assesses the precision of UAV operations, inspired by human peg-in-hole tasks and hummingbird dexterity in flight. The setup involves inserting a rod into an acrylic C-ring, with a 90 ° clamp angle to mimic human tests, as shown in Figure 6. A baseless hole is used to eliminate interaction effects.

Reflective motion capture markers are placed on the pegboard, and Vicon Tracker data is processed via Simulink, as depicted in Figure 2a. Consistency is maintained by using the same peg across trials, with dimensions selected based on Orsag et al. [7]. The peg, 10 % of the UAV rotor diameter (1.5 cm), is chosen to generalize across different UAVs.

Five hole sizes, ranging from 2.5 to 6.5 cm, are used to ensure test generalizability, with peg clearances from 1 to 5 cm. The pegboard is designed with CAD and 3D printed, as shown in Figure 7. The proposed approach involves waypoints using the pegboard's location coordinates, which enable the octocopter to follow a trajectory autonomously. This allows the test to be more accessible and affordable.

The test assesses the UAV system's real-time dexterity, including mechanical design, control architecture, and sensor integration, similar to human and robotic arm assessments [5]. A weighted scoring metric emphasizes precision, with higher weights for smaller holes. The peg-in-hole score is the cumulative success rate for each hole, weighted by hole size.

Specific conditions refine the test for fully-actuated UAVs, ensuring horizontal thrust is required without pitching or rolling. One condition involves translating a distance at least D_Y over 5 s or less, scaled to twice the propeller diameter (Figure 6). A second condition sets an RMSE threshold of 5 ° for horizontal thrust motion. These parameters focus on fully-actuated UAV qualities, making the test sensitive to dexterity variations across UAVs. The peg-in-hole score is detailed in Equation 3.

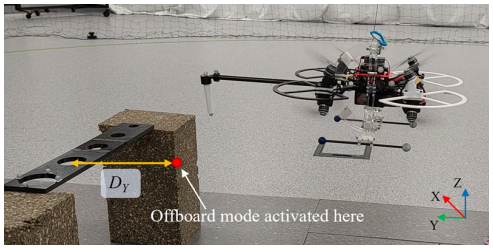


Figure 6: Illustration of conditions

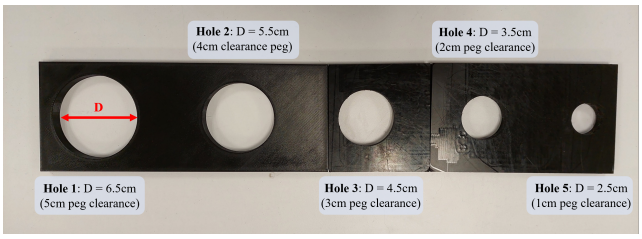


Figure 7: UAV pegboard designed for a 1.5 cm diameter peg

$$P = \sum_{1}^{5} w_u.S_u \tag{3}$$

where P is the peg-in-hole score, w_u is the weighting for peg-in-hole test, S_u is the success rate for peg-in-hole test, and u is 1, 2, 3, 4, or 5 corresponding to the pegboard hole.

From this equation it can be seen that the maximum score possible is $1 \times 100\,\% + 2 \times 100\,\% + 3 \times 100\,\% + 4 \times 100\,\% + 5 \times 100\,\% = 15$.

4.3.2 Experiments

The experiment commences by having a pegboard between two foam blocks before placing four markers around hole 1. The difference in X-axis (horizontal) displacement between each hole is added to the setpoint for the other pegboard holes. Subsequently, an object representing the UAV and the pegboard itself is created in the Vicon tracker. Assumptions regarding tuning are that the UAV being tested is tuned for free flight.

Furthermore, a marker is placed on the bottom of the peg to enable the peg's position using the Vicon tracker system to be located. This information is used in the Simulink interface to integrate the difference in the Y-axis pegboard location and peg location into offboard setpoints.

Upon completion of the setpoint configuration, the UAV is manually armed and transitioned into offboard mode, initiating autonomous navigation towards the designated hole. Following this, a stabilization period of at least two seconds is allowed before providing a Z-axis setpoint.

The setup is monitored throughout the experiment via a webcam, as depicted in Figure 2a. If the UAV deviates from its intended setpoints and makes contact with the pegboard, corrective measures are conducted to reposition the UAV.

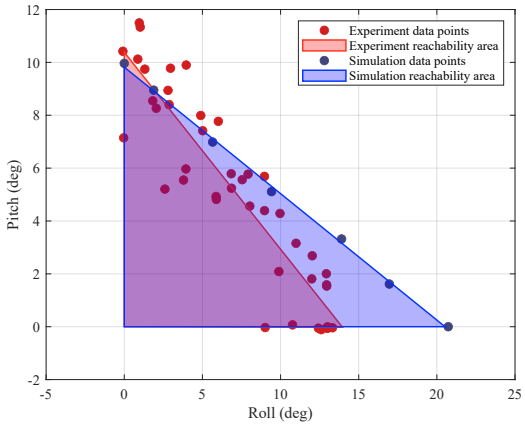


Figure 8: UAV Experiment vs Simulation Reachability Plot

5 RESULTS AND DISCUSSION

5.1 *Octocopter UAV Reachability*

Multiple experiments are conducted for each roll angle and a pitch limit found to form a cloud of points. A line of best fit is plotted in MATLAB to create the reachable workspace area. The octocopter in this study achieved an area of 71^{o^2} for the quarter reachability workspace illustrated in Figure 8 and consequently a dexterity score of: $\frac{4 \times 71.28}{360^2} \times 100\,\% = 0.22\,\%$. Comparatively, the simulation for this UAV using a quarter of the data achieving an area of 100^{o^2}, and a dexterity score of: $\frac{4 \times 100.44}{360^2} \times 100\,\% = 0.31\,\%$.

The results from this reachability test experiment provide an interpretation of the UAV's workspace that it can reach independently of a complex DoF arm. The experimental results closely resemble a simulated reachability dataset for the case study octocopter. As expected, the simulation's rolling capability is much higher, given that various factors in the experiment could reduce the UAV's thrust capability, such as motor heating and battery age. Interestingly, the experimental data for the simulation in the pitch limits is higher. This discrepancy is unexpected and could be attributed to the uncertainty in experimental data due to uncontrollable factors.

5.2 *Octocopter UAV Contact Test*

The case study octocopter is capable of successful contact for each angle assessed. The force data obtained calculates the average force exerted over the 30 s compression period. Moreover, the UAV maintained an attitude where the RMSE for all tests remained below 10 °.

Utilizing the collected data and Equation 2, a contact testing score of 1.01 is calculated for the octocopter weighing 1275 g. These results demonstrate the UAV's capability to sustain a force ranging from 2 to 3 N, while utilizing up to 80

http://www.imavs.org/

% or more of its thrust capacity for an extended duration.

The experiments and subsequent findings present a method for assessing UAV dexterity, highlighting the fully-actuated UAV's ability to withstand forces and maintain stable hovering during aerial manipulation tasks.

Understanding this score can advance UAV development for tasks such as window cleaning, where sustained force application is essential. For example, the window cleaning aerial vehicle in [23] requires different force closure requirements than that of a transmission line inspection UAV such as in [24]. This is because the typical window cleaning action requires a larger force normal to the window surface, as opposed to a transmission line application requiring a smaller force, at potentially different angles.

5.3 Octocopter UAV Peg-in-hole

The peg-in-hole test illustrated in Figure 6 consisted of five attempts for each peg angle, resulting in five success rates. The octocopter achieved success rates of 40 % for holes 1 and 2, a 20 % success rate for hole 3, and 0 % success rates for holes 4 and 5. Using Equation 3 the peg-in-hole score is calculated as: $1{\times}40\%{+}2{\times}40\%{+}3{\times}20\%{+}4{\times}0\%{+}5{\times}0\% = 1.8$. This score corresponds to 12 % of the maximum possible score of 15 (refer to Section 4.3.1).

The Y (forward) displacement from the pegboard at the point of offboard transition exceeds the required 0.3 meters for this case study, given that the octocopter's rotor diameter is 0.15 m. Therefore, the offboard PC sends a setpoint to the UAV to move a distance towards the pegboard of at least 0.3 m. Moreover, the time to cover this displacement is less than 5 s while maintaining roll, pitch, and yaw angles at approximately 0 °. These results and the attitude for the other successful attempts had low RMSE errors, indicating that the UAV is exhibiting control over six DoF since it can translate without changing attitude. If an under-actuated UAV undergoes the same tests, the attitude RMSE is likely to be significant, and therefore this would not be considered a successful attempt.

As expected, the success rate reduces with reduced hole sizes. The score of 1.8 is 12 % of the maximum score achievable, suggesting that the test can discriminate preciseness among different UAVs, however, this can only be verified once more than one UAV is assessed.

6 CONCLUSION

This study involved designing a standardised dexterity assessment for fully-actuated UAVs and evaluating the assessment's feasibility through experiments. Previously, it was difficult to identify how well a UAV would behave during aerial manipulation. This study introduced a novel perspective to the conventional analysis of UAVs, exploring a concept infrequently used within UAV research.

A definition of dexterity was created, and three comprehensive tests were designed. Furthermore, a score was created for each test, allowing the quantification and comparison across UAVs. Following the design of the tests, an in-depth experimental investigation was conducted that demonstrated how these tests would proceed with a fully-actuated octocopter UAV.

This research holds significance within the UAV community by presenting a benchmarking method for quantifying dexterity, thereby improving the assessment of aerial manipulation capabilities in UAVs. This methodology aids researchers in selecting UAVs best suited for various aerial manipulation applications, including window cleaning, sensing tasks, and powerline testing.

Future work includes implementing the tests on more UAVs, assessing the dexterity scores across UAVs, potentially developing an overall dexterity score, and demonstrating sensitivity in scores such that different UAVs could be rated on their dexterity and suitability to particular aerial manipulation applications. Furthermore, repeating these tests in different environments, such as with wind disturbances, could provide valuable insights into the dexterity of UAVs.

ACKNOWLEDGMENT

The research reported in this article was conducted as part of "Enabling unmanned aerial vehicles (UAVs) to use tools in complex dynamic environments "UOCX2104", which is funded by the New Zealand Ministry of Business, Innovation and Employment.

REFERENCES

[1] Anibal Ollero, Guillermo Heredia, Antonio Franchi, Gianluca Antonelli, Konstantin Kondak, Alberto Sanfeliu, Antidio Viguria, J Ramiro Martinez-de Dios, Francesco Pierri, Juan Cortés, et al. The aeroarms project: Aerial robots with advanced manipulation capabilities for inspection and maintenance. *IEEE Robotics & Automation Magazine*, 25(4):12–23, 2018.

[2] Yinshuai Sun, Zhongliang Jing, Peng Dong, Jianzhe Huang, Wujun Chen, and Henry Leung. A switchable unmanned aerial manipulator system for window-cleaning robot installation. *IEEE Robotics and Automation Letters*, 6(2):3483–3490, 2021.

[3] Guangying Jiang, Richard M. Voyles, and Jae Jung Choi. Precision fully-actuated uav for visual and physical inspection of structures for nuclear decommissioning and search and rescue. In *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–7, 2018.

[4] Raymond R. Ma and Aaron M. Dollar. On dexterity and dexterous manipulation. In *2011 15th International Conference on Advanced Robotics (ICAR)*, pages 1–7, 2011.

[5] Nathan Elangovan, Geng Gao, Che-Ming Chang, and Minas Liarokapis. A modular, accessible, affordable

http://www.imavs.org/

dexterity test for evaluating the grasping and manipulation capabilities of robotic grippers and hands. In *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 304–310, 2020.

[6] Joe Falco, Jeremy Marvel, and Elena Messina. *A roadmap to progress measurement science in robot dexterity and manipulation*. US Department of Commerce, National Institute of Standards and Technology, 2014.

[7] Matko Orsag, Christopher Korpela, Stjepan Bogdan, and Paul Oh. Dexterous aerial robots - mobile manipulation using unmanned aerial systems. *IEEE Transactions on Robotics*, 33(6):1453–1466, 2017.

[8] Katie E. Yancosek and Dana Howell. A narrative review of dexterity assessments. *Journal of Hand Therapy*, 22(3):258–270, 2009.

[9] Peter Feys, Ilse Lamers, Gordon Francis, Ralph Benedict, Glenn Phillips, Nicholas LaRocca, Lynn D Hudson, Richard Rudick, and Multiple Sclerosis Outcome Assessments Consortium. The nine-hole peg test as a manual dexterity performance measure for multiple sclerosis. *Multiple Sclerosis Journal*, 23(5):711–720, 2017.

[10] E. Rimon and J. Burdick. On force and form closure for multiple finger grasps. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 2, pages 1795–1800 vol.2, 1996.

[11] Alejandro Suarez, Victor M. Vega, Manuel Fernandez, Guillermo Heredia, and Anibal Ollero. Benchmarks for aerial manipulation. *IEEE Robotics and Automation Letters*, 5(2):2650–2657, 2020.

[12] Guangying Jiang and Richard Voyles. Hexrotor uav platform enabling dextrous interaction with structures-flight test. In *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–6, 2013.

[13] Mina Kamel, Sebastian Verling, Omar Elkhatib, Christian Sprecher, Paula Wulkop, Zachary Taylor, Roland Siegwart, and Igor Gilitschenski. The voliro omniorientational hexacopter: An agile and maneuverable tiltable-rotor aerial vehicle. *IEEE Robotics Automation Magazine*, 25(4):34–44, 2018.

[14] Guangying Jiang and Richard M. Voyles. *Dexterous UAVs for Precision Low-Altitude Flight*, pages 207–237. Springer Netherlands, Dordrecht, 2015.

[15] Jon Verbeke and Joris De Schutter. Experimental maneuverability and agility quantification for rotary unmanned aerial vehicle. *International Journal of Micro Air Vehicles*, 10(1):3–11, 2018.

[16] Hamza Mehmood, Takuma Nakamura, and Eric N. Johnson. A maneuverability analysis of a novel hexarotor uav concept. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 437–446, 2016.

[17] Mohammad Nasirul Haque, Bo Cheng, Bret W Tobalske, and Haoxiang Luo. Hummingbirds use wing inertial effects to improve manoeuvrability. *Journal of the Royal Society Interface*, 20(207):20230229, 2023.

[18] Pedro Henrique Mendes Souza. *Dynamics and Control of Unmanned Aerial Vehicles for Prolonged Interaction with Compliant Environments*. PhD thesis, ResearchSpace@ Auckland, 2023.

[19] Marco Tognon, Hermes A. Tello Chávez, Enrico Gasparin, Quentin Sablé, Davide Bicego, Anthony Mallet, Marc Lany, Gilles Santi, Bernard Revaz, Juan Cortés, and Antonio Franchi. A truly-redundant aerial manipulator system with application to push-and-slide inspection in industrial plants. *IEEE Robotics and Automation Letters*, 4(2):1846–1851, 2019.

[20] Robert Watson. *Unmanned Aerial Vehicles for Contact-Based Inspection*. PhD thesis, University of Strathclyde, 2023.

[21] Takahiro Ikeda, Shogo Yasui, Motoharu Fujihara, Kenichi Ohara, Satoshi Ashizawa, Akihiko Ichikawa, Akihisa Okino, Takeo Oomichi, and Toshio Fukuda. Wall contact by octo-rotor uav with one dof manipulator for bridge inspection. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5122–5127. IEEE, 2017.

[22] Karen Bodie, Maximilian Brunner, Michael Pantic, Stefan Walser, Patrick Pfändler, Ueli Angst, Roland Siegwart, and Juan Nieto. Active interaction force control for contact-based inspection with a fully actuated aerial vehicle. *IEEE Transactions on Robotics*, 37(3):709–722, 2021.

[23] Albert Albers, Simon Trautmann, Thomas Howard, Trong Anh Nguyen, Markus Frietsch, and Christian Sauter. Semi-autonomous flying robot for physical interaction with environment. In *2010 IEEE Conference on Robotics, Automation and Mechatronics*, pages 441–446, 2010.

[24] Quanwei Du, Wuzhong Dong, Wei Su, and Qi Wang. Uav inspection technology and application of transmission line. In *2022 IEEE 5th International Conference on Information Systems and Computer Aided Education (ICISCAE)*, pages 594–597. IEEE, 2022.

http://www.imavs.org/

Design of a Tapered-Wing Tiltrotor UAV for Multi-Mission Applications

Edgar U. Rojo-Rodriguez[1]*, Erik G. Rojo-Rodriguez[1], Sergio Araujo-Estrada[2], and Octavio Garcia-Salazar[1]
[1]Aerospace Engineering Research and Innovation Center, Faculty of Mechanical and Electrical Engineering,
Autonomous University of Nuevo Leon, Apodaca NL, Mexico.
[2]Department of Aeronautical and Astronautical Engineering,
University of Southampton, Southampton SO16 7PX, U.K.

http://www.imavs.org/

ABSTRACT

This paper outlines the design, development, and flight testing of a novel convertible aircraft with a fixed-wing and dual-tiltrotor configuration. Addressing the limitations of traditional Vertical Take-Off and Landing (VTOL) aircraft, the proposed design combines the efficient cruising of fixed-wing flight with the maneuverability of multirotor systems. Traditional VTOL aircraft face challenges such as limited range and payload capacity due to their dependence on rotorcraft technology, which is not optimized for high-speed travel and results in higher fuel consumption. The new design overcomes these issues by integrating a fixed wing with a dual-tiltrotor system. During take-off and landing, the tiltrotors provide vertical lift like a helicopter. In forward flight, the tiltrotors rotate to function as propellers, working with the fixed wing to enhance lift efficiency, thus improving cruise range and energy economy. The design process was supported by Computational Fluid Dynamics (CFD) and structural analysis to ensure optimal aerodynamic performance and structural integrity. A comprehensive flight-testing program validated the feasibility and performance of the design. Multiple tests were conducted covering all critical flight phases, including vertical take-off, hover, transition to forward flight, sustained cruise, and vertical landing. The collected flight data confirmed the design's improved maneuverability, efficiency, and controllability.

1 INTRODUCTION

The increasing need for efficient and versatile aircraft has driven ongoing innovation in vertical take-off and landing (VTOL) technologies, [1]. While traditional VTOL designs offer clear advantages in operational flexibility, they often face limitations in range, payload capacity, and cruise efficiency. This paper presents a novel approach to address these shortcomings by introducing a convertible tiltrotor aircraft,[2].

The increasing demand for ever-more capable unmanned aerial vehicles (UAVs) has driven significant advancements in vertical take-off and landing (VTOL) technologies. While traditional VTOL designs provide undeniable advantages in operational flexibility by enabling take-off and landing from confined spaces, they often face limitations in range, payload capacity, and efficiency during cruise flight. This trade-off between maneuverability and long-range capability restricts their applicability in various scenarios. [3]

To address these problems, researchers are exploring the exciting potential of convertible UAVs [3], [4], [5], [6], [7]. These innovative aircraft can transform their configuration mid-air, transitioning between vertical take-off/landing like a helicopter and efficient forward flight like a fixed-wing aircraft. This adaptability unlocks a wider range of missions by combining the:

- Vertical agility of multirotor UAVs for hovering, precision maneuvers, and operation in confined areas.
- High-speed, long-range flight capabilities of fixed-wing UAVs for efficient travel and covering larger distances.

A prominent approach within convertible UAVs is the tiltrotor design. Here, rotors mounted on the fuselage can tilt their orientation. In vertical flight mode, the rotors point upwards, generating thrust for hovering and maneuvering. During the transition, the rotors progressively tilt forward, converting their thrust into forward propulsion for efficient cruise flight. While tiltrotor designs offer a promising solution, there are ongoing research efforts to address challenges associated with:

- Complex control systems: Ensuring smooth and efficient transitions between flight modes requires sophisticated control algorithms to manage rotor tilt, thrust vectoring, and overall vehicle dynamics.

*Email address: edgar.rojordrg@uanl.edu.mx

http://www.imavs.org/

- Increased mechanical complexity: Tiltrotor mechanisms add weight and complexity to the design compared to simpler multi-rotor or fixed-wing UAVs.

- Aerodynamic efficiency: Optimizing the design for both hovering and forward flight can present trade-offs in terms of aerodynamic efficiency.

Current approaches include different configurations of UAVs that involve each author using their perspective to develop a rotor-actuator configuration in order to develop a workbench that is capable of making hybrid missions. Some of them are listed below:

- A prevalent configuration within the realm of convertible UAVs involves a quadrotor design equipped with a tilting mechanism, as addressed in [8]. This mechanism grants the ability to dynamically alter the direction of propulsion. Such configurations play a crucial role in investigating the behavior of tilting mechanisms and their significance for the broader field of convertible UAV research.

- A prototype employing vectorized thrust is presented in Oliveira et al., enabling the capability for motion without the need for corresponding body movement. This research holds particular importance in the realm of mechanism and behavior identification due to the methodological approach employed and the successful validation of effectiveness at this particular scale [9].

- The research explores the application of a tail-sitter configuration for convertible UAVs, expanding the diversity of model configurations, as presented by Flores et al.. This configuration is investigated alongside another convertible aircraft design, with the aim of evaluating the potential for different models to enhance efficiency in specific tasks [10].

- Zhao et al. work delves into the structural domain of deformable quadrotors. This field represents a distinct area of research compared to convertible UAV design, yet it holds significant relevance. This information is of vital importance to encompass the behavior of the aircraft and its effects [11].

- Bronz et al. contribution to the field of convertible UAVs lies in the exploration of a configuration centered around a fixed wing. This design approach prioritizes simplicity and functionality. The fixed-wing element provides inherent lift capabilities during forward flight, similar to conventional airplanes [4].

- Fayez et al. research builds on this concept by prioritizing long-endurance capabilities. This approach aligns with the need for UAVs capable of undertaking long-range missions. Fayez et al. delves into optimizing these fixed-wing convertible UAVs for extended flight times, potentially unlocking a new generation of convertible UAVs with transformative implications for long-range applications [12].

1.1 Convention and approach

This proposal builds upon the concept of a convertible aircraft design to explore a variation with unique features for superior performance. The symmetrical dual-tiltrotor configuration is avoided by introducing independent tilting mechanisms for the frontal rotors. This design grants the aircraft increased agility and maneuverability during hovering and low-speed flight, allowing for more precise control during critical phases such as search-and-rescue operations or navigating confined spaces.

Moreover, the design features a full fuselage with an FX-63 airfoil and a 40cm wingspan. This optimized fuselage profile is anticipated to significantly improve airflow, potentially reducing drag during forward flight. The FX-63 airfoil, renowned for its balanced lift and drag characteristics, is specifically selected to enhance overall aerodynamic efficiency, as demonstrated in our velocity distribution over the airfoil, as studied in figure ef CFDdis. The quadrotor configuration places two rotors on the wingtips for additional lift and thrust during forward flight. The remaining two rotors are situated on the bottom portion of the fuselage and function differentially, providing precise directional control, particularly during low-speed maneuvers and hovering. This configuration aims to achieve a balance between the stability of a conventional quad-copter and the directional controllability offered by a variable-thrust vectoring system.

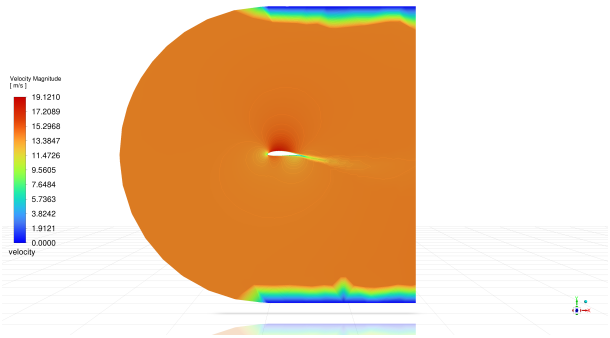


Figure 1: Airflow Distribution.

By incorporating these design elements, we intend to explore the potential for enhanced maneuverability, improved aerodynamic efficiency, and overall controllability within the designated flight envelope. The independent tilting frontal rotors, FX-63 airfoil fuselage, and differential thrust bottom rotors collectively represent a unique approach to convertible

http://www.imavs.org/

aircraft design, warranting further investigation through simulations and flight testing.

2 CONCEPTUAL DESIGN AND OPTIMIZATION

Convertible UAVs, offering capabilities on both vertical flight and cruise flight, present unique design challenges. This section delves into the key considerations and optimization strategies for conceptualizing an effective convertible UAV. The selected methodology outlines the subsequent steps to develop the aircraft design:

- Configuration selection: Identifying the most suitable configuration is crucial. This involves evaluating different configurations, such as tiltrotors, tail-sitters, and quad-rotor designs with tilting mechanisms. Each configuration has distinct advantages and limitations that must be considered based on the intended mission profile and operational requirements.

- Design parameters and optimization: Defining critical design parameters, such as wing span, rotor size, and propulsion system characteristics, is essential. Optimization techniques, including Computational Fluid Dynamics (CFD), structural analysis (FEA), and Solid Isotropic Material with Penalization method (SIMP) for optimization, are employed to refine these parameters, ensuring optimal aerodynamic performance and structural integrity.

- Performance analysis and trade-off studies: Conducting detailed performance analyses help assess various trade-offs, such as balancing VTOL efficiency with fixed-wing cruise performance. Trade-off studies are vital for understanding the implications of design choices on range, payload capacity, fuel efficiency, and overall aircraft performance.

- Mission profile integration: Integrating the specific mission profile into the design process ensures that the UAV meets operational requirements. This involves tailoring the design to support various missions, including long-distance cargo delivery, search and rescue operations, and infrastructure inspection.

- Design for manufacturability and maintainability: Ensuring the UAV design is manufacturable and maintainable is critical for practical implementation. This includes selecting materials and components that are cost-effective, readily available, and easy to assemble and maintain. Design strategies should prioritize ease of production and long-term reliability.

By addressing these fundamental considerations, the conceptual design and optimization process aims to develop a convertible UAV that meets the dual requirements of rotary wing and fixed-wing flight, maximizing operational efficiency and versatility. sectionIterative Design Process and Optimization Techniques

The design and optimization of convertible UAVs involve a highly iterative process where simulations and analyses play a central role. The use of Computational Fluid Dynamics (CFD) and Finite Element Method (FEM) analyses are particularly crucial in creating iterative design improvements.

2.1 CFD and FEM in iterative design

- Computational Fluid Dynamics (CFD): CFD simulations are used extensively to analyze the aerodynamic properties of the UAV design. By simulating airflow around the aircraft, CFD helps CFD helps identify and mitigate potential issues such as drag, turbulence, and flow separation. This iterative process involves running multiple simulations, adjusting design parameters, and re-evaluating performance until optimal aerodynamic efficiency is achieved.

- Finite Element Method (FEM): FEM analysis is employed to assess the structural integrity of the UAV. This technique involves creating a detailed model of the aircraft's structure and subjecting it to various stress and load conditions. FEM helps in identifying weak points and areas prone to deformation or failure. Iterative FEM analyses allow for continuous refinement of the structural design, ensuring robustness and durability.

It was carried FEM study to estimate deformation; the most important one was made for ABS as the main structure and carbon fiber tubes for external beams, as seen at 2, with a maximum of 0.9258mm, considering max force estimate by rotors of 200gr each, and fixed at the center of gravity, condition for hover flight. For a fixed-wing based flight, small

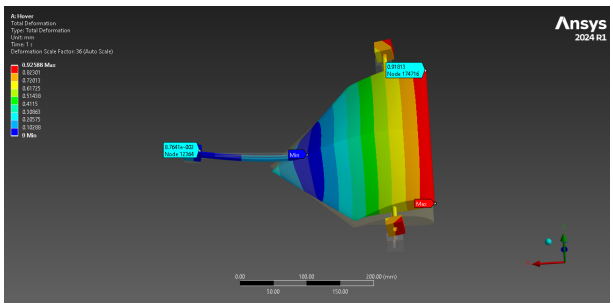


Figure 2: Structure deformation on hover condition.

forces are considered on rotors for propulsion and bottom rotors just to balance, as seen in figure 3. Preliminary results shows that cruise flight is less demanding to the structure, considering a distributed force on the wing of same value of structural mass, for this case 4.41N.

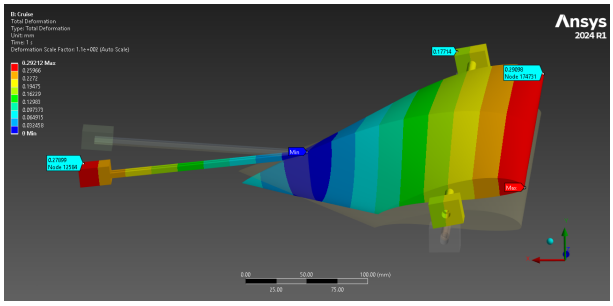


Figure 3: Structure deformation on cruise condition.

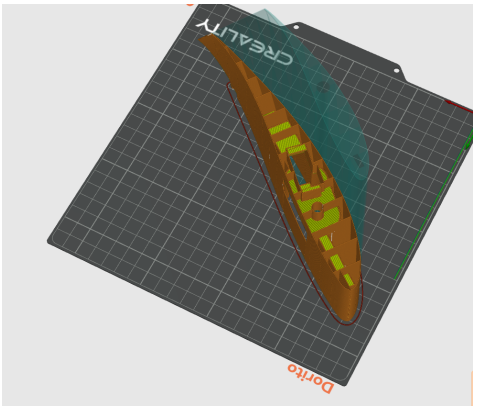


Figure 4: Internal 3D printed Wing structure.

2.2 *3D Printing of internal wing structure*

Advanced optimization techniques, such as those mentioned above, are also applied in the design and fabrication of the internal structure of the UAV wing using 3D printing technology. By integrating CFD for forces distribution on flight and FEM analyses for deformations, stress, and strain are carried out, the internal wing structure is optimized for both aerodynamic performance and structural integrity. This process involves the following steps:

- Design Optimization: Using CFD and FEM analyses, the internal wing structure is designed to achieve an optimal balance between strength and weight. The iterative optimization process ensures that the design can withstand aerodynamic forces while maintaining minimal weight, as shown in figure 4.
- Material Selection: Selecting appropriate materials for 3D printing is critical. Materials are chosen based on their mechanical properties, such as strength, flexibility, and weight, to ensure they meet the design specifications.
- Additive Manufacturing: The optimized internal structure is fabricated using 3D printing technology. This allows for complex geometries that would be difficult or impossible to achieve with traditional manufacturing methods. Additive manufacturing enables the production of lightweight, strong, and intricate structures tailored to the specific aerodynamic and structural requirements of the UAV.
- Testing and Validation: The 3D printed wing structures are subjected to rigorous testing to validate their performance. This includes static load tests, fatigue tests, and in-flight testing to ensure that the structures meet all safety and performance criteria.

By leveraging these advanced simulation tools, optimization techniques, and 3D printing technologies, the iterative design process of convertible UAVs ensures the development of highly efficient, reliable, and mission-capable aircraft.

In the development of lightweight structures for prototypes, the choice of materials plays a critical role in determining performance, durability, and suitability for specific applications. Experimental testing of materials such as PLA, ABS, PETG, Nylon, and PP provides valuable insights into their deformation characteristics and mechanical properties under load; important data are carried out in table 1.

The deformation and stress data obtained from these tests are essential for evaluating how each material performs when integrated into lightweight structures. For instance, PLA exhibits relatively low deformation and moderate stress resistance, making it suitable for applications where weight reduction is critical but without extreme mechanical demands. On the other hand, Nylon shows higher deformation but offers excellent toughness, making it suitable for applications requiring impact resistance.

Among the tested materials, ABS stands out due to its equilibrium between deformation, stress resistance, and thermal stability. ABS demonstrates a balance of strength and flexibility, making it well-suited for lightweight structures where structural integrity under varying loads is essential. Moreover, ABS thermal resistance ensures stability in diverse environmental conditions, crucial for applications exposed to fluctuating temperatures.

Based on these findings, ABS was selected for our prototype's lightweight internal lattice structures. Its combination of mechanical properties and thermal stability aligns with the operational requirements, ensuring reliable performance under real-world conditions. This informed decision underscores the importance of rigorous material testing in optimizing prototype designs for maximum efficiency and durability.

By leveraging material testing data, engineers can make informed decisions to enhance the structural integrity and performance of lightweight prototypes, paving the way for innovative solutions in various industries, from aerospace to automotive and beyond.

http://www.imavs.org/

Table 1: Comparison of Deformation and Stress of Lightweight Structures

Material	Deformation (%)	Stress (MPa)
PLA	1.50	51.20
ABS	2.20	39.80
PETG	1.80	45.35
Nylon	3.50	33.07
PP	2.80	25.25



Figure 5: Multiple techniques used for Wing.

2.3 Prototyping and final design

The optimization techniques mentioned above were used to develop a series of 26 prototypes in order to test and refine the UAV design. The process is illustrated in Figure 5, which shows multiple iterations and the final wing development. Each prototype incorporated feedback from previous iterations, allowing for continuous improvements in performance, efficiency, and structural integrity. The final prototype was manufactured using Acrylonitrile Butadiene Styrene (ABS) material, chosen for its durability and lightweight properties. Additionally, the internal structure of the wing featured lightweight lattice structures that were 3D printed to optimize strength while minimizing weight.

- Prototyping Process:The iterative prototyping process involved creating and testing 26 different prototypes. Each iteration was subjected to rigorous CFD and FEM analyses, as well as practical flight tests, to identify and address any design issues.
- Final Prototype Design: The final prototype incorporated lessons learned from all previous iterations. It was constructed using ABS material for the external shell, providing a balance of strength and weight. The internal wing structure utilized advanced 3D printed lattice structures, which were optimized through FEM analysis to ensure maximum structural integrity with minimal material usage.
- Testing and Validation: The final prototype underwent comprehensive testing, including static load tests, dynamic flight tests, and endurance trials. These tests validated the design's performance, confirming that it met all operational requirements and performance benchmarks.

3 FLIGHT CONTROL SYSTEM DESIGN

Convertible UAVs, due to their ability to switch between rotary wing and fixed-wing flights, require a sophisticated flight control system (FCS) to ensure smooth transitions and stable flight across all regimes.
The flight control system (FCS) architecture for a convertible UAV plays a critical role in ensuring seamless transitions and stable flight across rotary wing and fixed-wing regimes. This architecture typically comprises a central Flight Controller (FC) unit responsible for real-time data processing, control algorithm execution, and actuator command generation. Various sensors, including an Inertial Measurement Unit (IMU) for orientation and acceleration, a Global Positioning System (GPS) for positioning, an airspeed sensor for wind speed and direction, and motor encoders for precise thrust and tilt angle control (in tilting rotor/duct configurations), provide essential feedback to the FC. Actuators, such as servo motors for control surface movement and Electronic Speed Controllers (ESCs) for electric motor control, translate electrical commands from the FC into physical actions on the UAV.
The software architecture leverages a modular design, separating functionalities like sensor fusion for unified state estimation, flight mode control for managing transitions and activating specific algorithms, stabilization control for real-time adjustments based on sensor data, navigation control for path planning and execution, and health monitoring for system diagnostics. Communication protocols facilitate data exchange and control commands between modules and the ground control station (GCS). Redundancy in critical components and fail-safe mechanisms enhance system robustness and ensure safe operation in case of potential sensor or actuator malfunctions. This well-defined system architecture empowers the FCS to effectively manage the complexities of convertible UAV flights and maintain optimal performance across diverse flight modes.
In the context of convertible UAVs, where transitions between VTOL and fixed-wing flight modes demand rapid adaptation of control strategies, the ESP32 capabilities are particularly advantageous. Its multicore capabilities and robust sensor fusion processing, implemented through C++ programming, empower UAV systems with enhanced real-time control, agility in flight mode transitions, and reliability in adverse conditions. Future research may explore further optimizations and applications of the ESP32 platform to expand the operational capabilities and resilience of convertible UAVs in diverse environments.
Our custom flight computer allows us to monitor and design all the interactions with our system and include model-based control schemes, as shown in figure 6. Many communications

http://www.imavs.org/

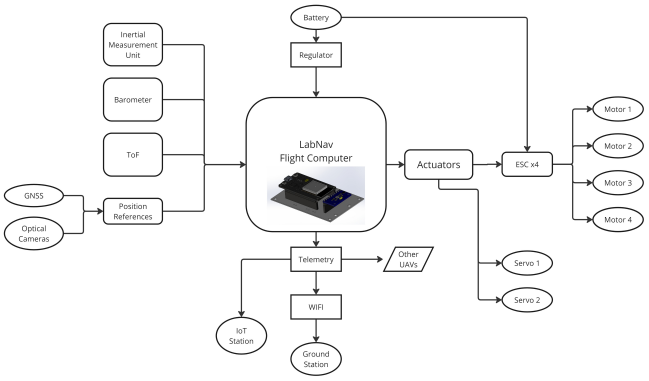


Figure 6: Sensor capabilities Diagram.

and implementations are allowed.

Control scheme for this continuous convertible UAV is devolved at others works, for this case is only presented main scheme on figure 7

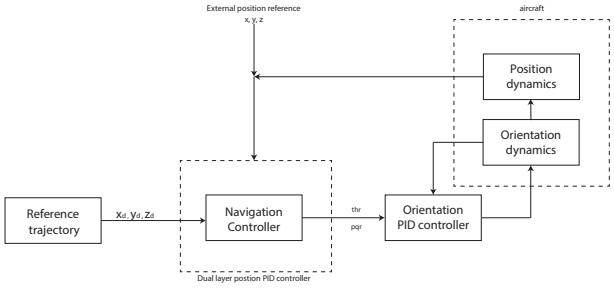


Figure 7: Control Scheme.

4 FLIGHT-TESTING PROGRAM

Convertible UAVs, with their rotary wing and fixed-wing flight capabilities, necessitate a rigorous flight-testing program to validate performance, stability, and safety across diverse flight regimes. This program typically follows distinct phases. Pre-flight testing commences with ground testing, meticulously examining the entire system (propulsion, flight control system (FCS), communication) through static motor tests, control surface deflection checks, and simulated flight scenarios. The validation of the system employs a comprehensive design of experiments (DoE) approach, enabling thorough testing of the system's capabilities. The chosen experiments focus on tracking missions, facilitating the assessment of the system's performance under consistent patterns and diverse movement combinations. It is crucial to note that each mission is tailored to evaluate the system performance across various scenarios that may not inherently support convertibility due to constraints related to tracking position. These missions provide valuable insights into the system's adaptability and performance under different operational contexts.

The Navigation Laboratory, at the Aerospace Engineering Research and Innovation Center, Faculty of Mechanical and Electrical Engineering, Autonomous University of Nuevo Leon, features a tracking system based on 16 VICON T-40 cameras, which obtain the information processing at a ground station to estimate general angular and linear position states. Figure 8 depicts our platform, where the system is distributed across a tracking station that transmits data to the network and communicates with the UAV via a secondary network of individual computers.



Figure 8: VICON Tracking system.

4.1 Vertical take-off

Vertical take-off capabilities of convertible UAVs involve crucial experiments to ensure optimal performance. Comprehensive testing verifies the efficiency and stability of vertical ascent and descent, including:

- Power and Thrust Testing: Evaluating propulsion systems under varying load conditions to validate thrust capabilities.
- Stability Assessments: Testing control algorithms and flight dynamics to ensure steady vertical flight.

The circular trajectory chosen measures 1750 mm in radius, with a height of 1800 mm and a velocity of 0.03 rev/s. This trajectory assessment emphasizes the tilting mechanism essential for achieving such motion without directly manipulating the rotors; instead, they are controlled solely by the mixer for stabilization.

The trajectory is influenced by geometrical errors that are inherent to actuator resolution limitations. These limitations are visible in slight deviations observed in online trajectory representations. Despite these challenges, the trajectory closely follows the intended path, maintaining acceptable tracking accuracy with minimal drift.

Figure 9 compares the actual XYZ and ideal trajectories. A primary objective of this design is to minimize the pitch angle, mainly to avoid interference with wing-generated forces.

Figure 10 compares real versus ideal pitch angles, emphasizing minimal deviation (pitch-less) to maintain optimal flight characteristics. The system demonstrates excellent per-

http://www.imavs.org/

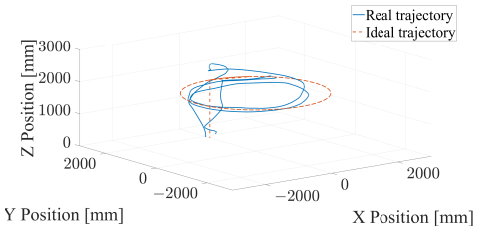


Figure 9: XYZ trajectory tracking, real vs ideal.

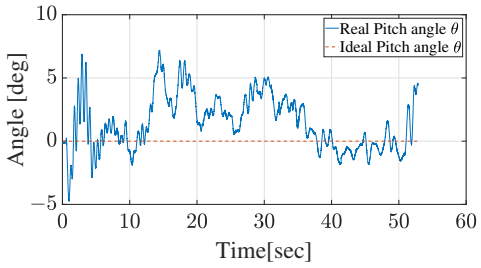


Figure 10: Pitch angle real vs ideal (pitch-less).

formance, maintaining a average error of 126.0183 mm and angular errors within ±3 degrees, as observed at table 4.1.

Axis	Average	Root Mean Squared	Mean Absolute	Maximum Absolute
X	80.0504	585.5509	489.9378	1.5494e+03
Y	126.0183	644.7709	465.6277	1.4141e+03

4.2 Transition

This experiment highlights the rapid transition capability of the system using tilting rotors, where the input involves swift changes in tilting angles.

Figure 11 illustrates the XYZ trajectory tracking, comparing real-time performance with the ideal trajectory. A linear trajectory is implemented in this scenario to facilitate the transition between hover and cruise flight modes.

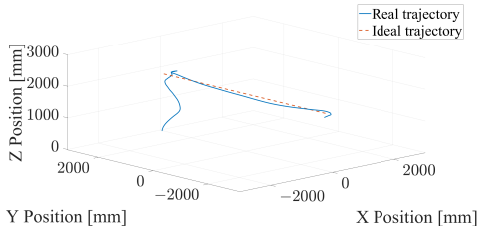


Figure 11: XYZ trajectory tracking, real vs ideal.

Figure 12 displays velocity development along the linear trajectory. During this brief experiment, the control scheme effectively compensates for altitude loss while accelerating the aircraft to a maximum speed of 2 m/s. It is observed that the system remains in hover mode during this acceleration,

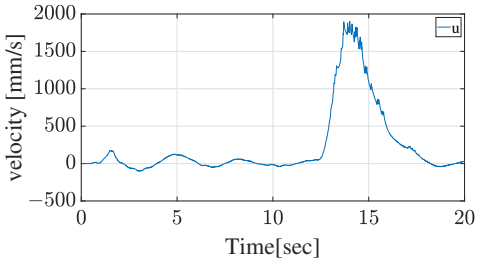


Figure 12: U velocity develop at Ramp

with the rotors not fully transitioning. Nonetheless, this short-duration experiment provides insights into expected dynamic behavior.

Figure 13 demonstrates the behavior of lift generated by the overall rotor forces, particularly notable between 8 to 13 seconds, where an increase in lift corresponds directly to a reduction in the required signal to maintain altitude.

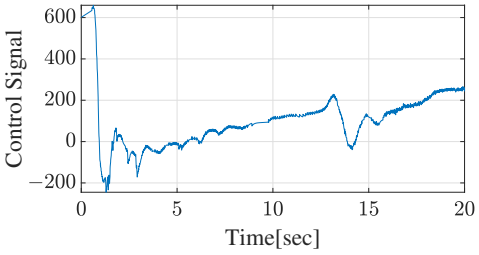


Figure 13: Lift behavior on global rotor forces.

As exemplified here, the transition phase of convertible UAVs underscores the system capability to swiftly and effectively switch between different flight modes, demonstrating crucial dynamics and control responses during such maneuvers.

Axis	Average	Root Mean Squared	Mean Absolute	Maximum Absolute
X	85.1773	232.5250	192.9673	536.5002
Y	407.0696	1.1919e+03	520.0588	4.9525e+03

Each phase contributes to validating the convertible UAV overall performance and safety across diverse flight regimes, ensuring that experimental results meet acceptable standards for operational deployment.

5 CONCLUSION

The presented work explores the innovative realm of convertible UAVs, specifically focusing on a novel tiltrotor design aimed at combining the agility of multirotor UAVs with the efficiency of fixed-wing aircraft. This approach addresses the inherent trade-offs in traditional VTOL designs by enabling seamless transitions between vertical take-off and landing and efficient forward flight. The paper emphasizes key challenges such as complex control systems, increased

http://www.imavs.org/

mechanical complexity, and aerodynamic efficiency, which are critical considerations in designing and optimizing such aircraft.

Through an iterative design process involving advanced simulation techniques like Computational Fluid Dynamics (CFD) and Finite Element Method (FEM) analysis, the authors developed and refined a prototype convertible UAV. This process included optimizing the aircraft's aerodynamic performance, structural integrity, and overall efficiency. Integrating 3D printing for internal wing structures and rigorous material testing underscored a commitment to lightweight yet durable design solutions.

The flight control system (FCS) design was another pivotal aspect, ensuring smooth transitions and stable flight across VTOL and fixed-wing modes. Leveraging sophisticated sensor fusion, modular software architecture, and robust actuators, the FCS demonstrated capabilities crucial for managing the complexities of convertible UAV flight.

A comprehensive flight-testing program validated the system's performance under various conditions, including vertical take-off, transitions between flight modes, and trajectory tracking. The experiments confirmed the aircraft's operational capabilities, stability, and safety, reinforcing the feasibility of the tiltrotor concept for diverse missions.

In conclusion, this work contributes significantly to convertible UAVs by proposing an innovative tiltrotor design supported by rigorous design methodologies, advanced simulation tools, and comprehensive testing protocols. The findings pave the way for further advancements in hybrid UAV technologies, promising enhanced operational flexibility and mission capabilities in civilian and military applications. The flight-testing program conducted rigorous experiments to validate the trajectory path following the capabilities of the convertible UAV, focusing on both vertical take-off scenarios and transitions between flight modes. For the vertical take-off phase, the UAV executed a circular trajectory with a radius of 1750 mm and a height of 1800 mm. The objective was to assess the aircraft ability to ascend and descend vertically while maintaining stability and accuracy in trajectory tracking. Using its tilting rotor mechanism, the transition experiment evaluated the UAV's capability to switch between hover and cruise flight modes. A linear trajectory was chosen to facilitate smooth transitions and assess dynamic responses during acceleration.

http://www.imavs.org/

ACKNOWLEDGEMENTS

The authors would like to thank the Aerospace Engineering Research and Innovation Center of the Faculty of Mechanical and Electrical Engineering at the Autonomous University of Nuevo Leon and the Department of Aeronautical and Astronautical Engineering at the University of Southampton for the facilities of this research.

REFERENCES

[1] Barnes Mccormick. *Aerodynamics of V/STOL flight*. Courier Corporation: North Chelmsford, MA, USA, 1999.

[2] Peter Germanowski, Brandon Stille, and Michael Strauss. Technology assessment for large vertical-lift transport tiltrotors. *NASA/CR–2010-216384*, 05 2010.

[3] Guillaume J.J. Ducard and Mike Allenspach. Review of designs and flight control techniques of hybrid and convertible vtol uavs. *Aerospace Science and Technology*, 118:107035, 11 2021.

[4] Murat Bronz, Ewoud J. Smeur, Hector Garcia de Marina, and Gautier Hattenberger. Development of a fixed-wing mini uav with transitioning flight capability. American Institute of Aeronautics and Astronautics, 6 2017.

[5] D. Aláez, X. Olaz, M. Prieto, J. Villadangos, and J. J. Astrain. Vtol uav digital twin for take-off, hovering and landing in different wind conditions. *Simulation Modelling Practice and Theory*, 123:102703, 2 2023.

[6] Adnan S. Saeed, Ahmad Bani Younes, Shafiqul Islam, Jorge Dias, Lakmal Seneviratne, and Guowei Cai. A review on the platform design, dynamic modeling and control of hybrid uavs. pages 806–815. IEEE, 6 2015.

[7] J. A. Bautista, A. Osorio, and R. Lozano. *Modeling and Analysis of a Tricopter/Flying-Wing Convertible UAV with Tilt-Rotors*. International Conference on Unmanned Aircraft Systems, Miami, FL, 1 edition, 2017.

[8] Gerardo Flores, Juan Escareno, R. Lozano, and Sergio Salazar. Quad-tilting rotor convertible mav : Modeling and real-time hover flight control. *Journal of Intelligent and Robotic Systems*, 65:457–471, 01 2012.

[9] Tomas Lopes de Oliveira, Andre Anglade, Tarek Hamel, and Claude Samson. Control of convertible uav with vectorized thrust. *IFAC-PapersOnLine*, 56:9288–9293, 2023.

[10] A. Flores, A. Montes De Oca, and G. Flores. A simple controller for the transition maneuver of a tail-sitter drone. *Proceedings of the IEEE Conference on Decision and Control*, 2018-December:4277–4281, 7 2018.

[11] Na Zhao, Yudong Luo, Hongbin Deng, and Yantao Shen. The deformable quad-rotor: Design, kinematics and dynamics characterization, and flight performance validation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2391–2396, 2017.

[12] Karim Fayez, Yuchen Leng, Thierry Jardin, Murat Bronz, and Jean-Marc Moschetta. Conceptual design for long-endurance convertible unmanned aerial system. American Institute of Aeronautics and Astronautics, 1 2021.

http://www.imavs.org/

Effect of Aerodynamic Summation on Multirotor UAV Hover Performance Estimates

N. J. Kay*, P. J. Richards, K. A. Stol,
Department of Mechanical and Mechatronics Engineering, University of Auckland, Auckland, New Zealand

R. D. Buchanan,
Cropsy Technologies, Auckland, New Zealand

J. X. J. Bannwarth
Seachange New Zealand Ltd, Auckland, New Zealand

http://www.imavs.org/

ABSTRACT

Aerodynamic forces are important for the development of multirotor Unmanned Aerial Vehicles, both in selecting hardware and designing controllers. In absence of aerodynamic data specific to a new design, developers turn to summation, adding loads of individual components to represent the whole aircraft. This neglects interaction between the components, a critical aspect for overactuated designs, which often have overlapped or tilted rotors. As such, this work looks at the errors induced by the summation method for a tilted-rotor octorotor. Force estimates from a summation method for steady wind conditions are first compared to a whole-aircraft model and wind tunnel data. Subsequently, the resulting simulated hover performance of each model in elevated winds is compared to wind tunnel free flight tests. Although the summation method tends to underestimate sensitivity to flow angle, it is generally suitable for predicting aircraft performance at lower wind speeds; error increasing with wind speed.

1 INTRODUCTION

Multirotor Unmanned Aerial Vehicles (UAVs) have gained popularity for aerial observation in both military and civilian contexts [1]. The attraction is their ability to operate anywhere without supporting infrastructure [2], and the ability to operate in proximity to obstacles, such as trees and buildings, too hazardous for piloted aircraft [3]. Overactuated aircraft are becoming popular for such operations, due to their enhanced station-holding capabilities while conducting a task [4].

During the development of these aircraft, it is common to employ component summation, adding the individual parts to represent the whole. Product data from off-the-shelf components, such as propellers, motors and batteries, form a database, used in optimisers such as those of Delbecq et al. [5], and Al-Zubaidi and Stol [6]. The net system performance is then estimated from extrapolation of this data. This, however, neglects aerodynamic interaction between components.

Aerodynamic interaction is configuration-dependent and cannot be gauged from databases. For example, when coaxial rotors are employed; the downwash from the upper rotor impinges on the lower rotor, and the inflow from the latter reduces the thrust of the former [7, 8]. Such interaction is also observed with partially-overlapped [9] and canted rotor pairs [4], common on overactuated designs. Interaction with the body also exists, particularly in the aerodynamic moments [10].

Aerodynamic interactions are compounded by wind conditions, which are often highly unsteady in UAV operating environments [11]. The forces on isolated rotors are known to be highly sensitive to wind speed and angle [12]. The forces on the body, and the interaction between rotor wakes, however, are more difficult to predict [13]. While high-fidelity CFD may provide more accurate aerodynamics, this is computationally costly, to the point of being prohibitive, for rapid aircraft development [14]. In absence of such aerodynamic data, the true performance of the aircraft is difficult to predict.

While the summation method is useful for obtaining initial aircraft performance estimates, its accuracy for an overactuated design in its operating environment has not been quantified. This is critical because of the greater component interaction typical of these designs (compared to a conventional quadrotor) and elevated wind environment in which they are intended to be used. This paper looks at the implications of the summation method on the aerodynamics forces acting on an overactuated octorotor, employing a canted-rotor design, and the consequent deviations in simulation-predicted hover performance.

2 METHODOLOGY

2.1 Aircraft

The aircraft for this work is a Canted-Rotor Planar Octorotor, developed by Chen et al. [15, 16]. The layout is shown in Figure 1, with eight rotors canted 31° towards its neighbour, about either the aircraft *X* (longitudinal) and *Y* (lateral) axis. There are eight two-blade T-MOTOR 6x2CF propellers (6 in diameter, 2 in pitch, carbon fibre), each driven by a T-MOTOR Navigator MN1806 2300Kv brushless DC motor. The main body has an approximately

* nicholas.kay@auckland.ac.nz

http://www.imavs.org/

cylindrical planform, with the rotors at a radius of 0.25 m from the centre of gravity.

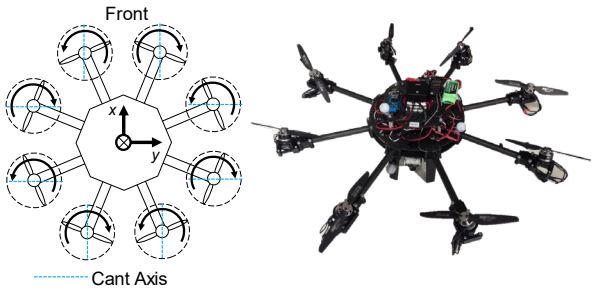


Figure 1: Canted Rotor Planar Octorotor

2.2 Aerodynamic Modelling

Wind tunnel testing was conducted on both the full aircraft (referred to as the "Whole Aircraft" case, with the body and all eight rotors installed), and the individual components (the body and a single rotor, each assessed individually). The test conditions are listed in Table 1. The Angle of Attack (AOA, α) is defined relative to the *X*-*Y* plane of the body for the Whole-Aircraft and Body-Only cases, and the rotor disk plane for the Rotor-Only test. This can be seen in Figure 2 for the whole-aircraft case. Thus, a negative AOA represents aircraft (or component) nose-down. The rotor speed was maintained as constant for each data collection, with all eight rotors in the whole-aircraft case operating at uniform speed. In all cases, the components were rigidly mounted on a JR3 six-axis loadcell, data being acquired for 30 s at 1000 Hz.

	Whole Aircraft	**Rotor**	**Centre Body**
Wind Speed, *U* (m s $^{-1}$)	0, 4, 6, 8, 10 m s^{-1}		
AOA, α (°)	-60° to +30°, 15° increments		
Rotor Angular Velocity, ω (rad s^{-1})	0, 737, 1317, 1742 (0, 25, 50, 75% maximum)		N/A

Table 1: Experimental conditions for aerodynamic force measurements

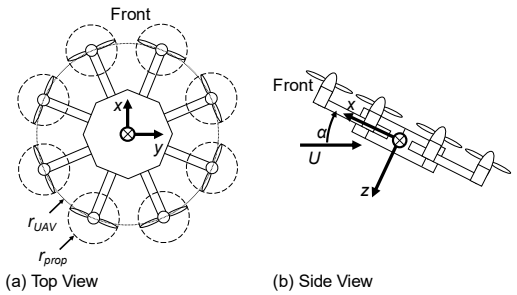


Figure 2: Coordinate System Definition (a) top view and (b) side view

An external, constant-voltage power supply was employed, providing 11.8V at the aircraft. Batteries were, however, fitted to account for their aerodynamics. The effects of blockage, wall interference and downwash impingement are considered negligible due to the size of the wind tunnel section (3.6 m wide, 2.5 m tall) relative to the components tested.

As data could not be acquired for every state likely to be simulated, a semi-empirical model was used to interpolate between the data points. The Bannwarth et al. [17] model was used, due to the ease of fitting experimental data to both the whole aircraft and individual components, as needed for summation.

This model fits curves to experimental data for the *Z*-axis force (F_z), *X*-axis force (F_x) and *Y*-axis moment (M_y) acting on the aircraft, as shown in Equations (1) to (3). The referenced radii are seen in Figure 2. The propeller radius, r_{prop}, is 0.0762 m while that of the airframe, r_{UAV}, is 0.25 m, resulting in circular reference areas A_{prop} of 0.018 m^2 and A_{UAV} of 0.196 m^2. The wind speed and rotor speed are U and ω, respectively, and air density is ρ. The C coefficients are empirically-fitted to the wind tunnel data for the whole-aircraft case, the process of which can be found in Bannwarth et al. [17]. The values used in this work may be found in the Appendix. As seen in the Appendix, these coefficients are dependent upon the Tip-Speed Ratio (TSR, λ) and the AOA. As Bannwarth et al. [17] assumed axisymmetric loading, the *Y*-axis force (F_y) and *X* and *Z* axis moments (M_x and M_z) are neglected. Note that the coefficients are all fitted from whole-aircraft tests, and assume that the n rotors are identical in geometry, speed and inflow angle.

$$F_z = C_{1,z}\frac{1}{2}\rho U^2 A_{UAV} + nC_{2,z}\frac{1}{2}\rho U\omega r_{prop}A_{prop} + nC_{3,z}\frac{1}{2}\rho\omega^2 r_{prop}^2 A_{prop} \quad (1)$$

$$F_x = C_{1,x}\frac{1}{2}\rho U^2 A_{UAV} + nC_{2,x}\frac{1}{2}\rho U\omega r_{prop}A_{prop} \quad (2)$$

$$M_y = C_{M1,y}\frac{1}{2}\rho U^2 A_{UAV} r_{UAV} + nC_{M2,y}\frac{1}{2}\rho U\omega r_{prop}^2 A_{prop} \quad (3)$$

For the summation method, the AOA, TSR and reference areas are unique to each body, with the coordinate frame attached to the individual body. The three C coefficients are then found individually for each component, and forces reconstructed via Equations (4) to (6). Note that r_{UAV} and r_{prop} are exchanged for r_i, and A_{UAV} and A_{prop} are replaced with A_i, where i designates the individual component. These forces and moments must be rotated and translated into the aircraft coordinate frame before being added to the aerodynamic loads of the other components. No models are included for aerodynamic interaction.

$$F_{z,i} = C_{1,z,i}\frac{1}{2}\rho U^2 A_i + C_{2,z,i}\frac{1}{2}\rho U\omega_i r_i A_i + C_{3,z,i}\frac{1}{2}\rho\omega_i^2 r_i^2 A_i \quad (4)$$

$$F_{x,i} = C_{1,x,i}\frac{1}{2}\rho U^2 A_i + C_{2,x,i}\frac{1}{2}\rho U\omega_i r_i A_i \quad (5)$$

$$M_{y,i} = C_{M1,y,i}\frac{1}{2}\rho U^2 A_i r_i + C_{M2,y,i}\frac{1}{2}\rho U\omega_i r_i^2 A_i \quad (6)$$

2.3 Flight Simulation

The flight simulator of Chen et al. [15, 16] was used to assess the effect of summation on hover estimates. The aircraft was given a constant position set-point, which simulated attempting to maintain that position in hover for 700 s. To remove initial simulation transients, the first 100 s of data was discarded prior to analysis. These simulations were run firstly using the whole-aircraft coefficient fits, then the summation method.

Figure 3 shows the basic form of the simulation, which was implemented in Matlab Simulink. This simulation separates the rotor and body aerodynamic loads, in the *Motor Model* and *Body Model* blocks, respectively, allowing for easy implementation of the summation method. It should be noted that the *Motor Model* block not only includes the propeller aerodynamics, but also the inertia of the motor-propeller unit and the electrical characteristics of the motor. Similarly, the *Aircraft Dynamics* block uses the inertia matrix measured on the aircraft as-built, and the *Position and Attitude Control Module* uses the as-built tuning.

For the whole-aircraft simulation, Equations (1) to (3) were implemented across both blocks with the associated whole-aircraft coefficients. However, rather than multiplying the C_2 and C_3 terms by n rotors, the forces for these terms were calculated n times, using the individual rotor speeds. This does not represent a summation model, however, as the coefficients were derived from the whole-aircraft data, thus including interaction, but neglecting local AOA. Simulations using the summation method implemented Equations (4) to (6) in both the *Motor Model* and *Body Model* blocks for each body, using the individual component coefficients found in the Appendix and accounting for the local AOA.

A limitation of the model is its quasi-steady aerodynamics, failing to account for transient delays in the pressure field. Furthermore, the onset flow did not deform over the airframe, as would be expected for flow passing over a body. However, this is consistent across both the summation and whole aircraft simulations.

Various unsteady wind conditions were employed, as listed in Table 2. This turbulent flow was recorded in the wind tunnel with an upstream turbulence grid and then used as an input timeseries of data. At speeds of 0, 5.6 and 12.8 m s^{-1}, Bannwarth [18] performed hover position hold tests with the same wind tunnel configuration and aircraft, providing a reference for the simulation.

Mean Wind Speed, U (m s^{-1})	0, 5.6, 7.4, 9.8, 12.8
Turbulence Intensity, I_u (%)	10

Table 2**:** Flow Conditions Simulated

3 RESULTS

3.1 Static Load Estimation

Before examining the free-flight performance, the steady force and moment estimates of the baseline model of Bannwarth et al. [17] are compared to the summation form. This is because any errors associated with the data fit carry through to the flight simulation. For the whole-aircraft case, the coefficients are derived from the wind tunnel data, and then the model predictions are found for the same cases. With the summation model, the coefficients were first found for each component from its individual wind tunnel data.

The overall error is summarised in Table 3 as a root-mean-square (RMS) of the error recorded across all cases. The R^2 value for each model is also given as an indication of the overall fit quality of both methods. In all cases, the summation model has greater error than the whole-aircraft case. This is not unexpected, as the whole-aircraft model inherently accounts for aerodynamic interaction between components, unlike the summation method. However, the overall error magnitude is comparable, and may allow for reasonable estimates via summation over portions of the test range.

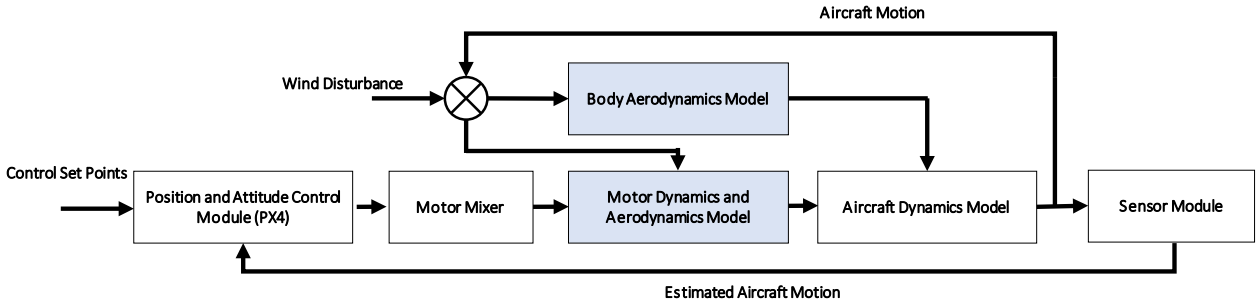


Figure 3: Flight simulation block diagram

http://www.imavs.org/

http://www.imavs.org/

	Whole Aircraft		Summation	
	RMS Error (N, Nm)	R^2	RMS Error (N, Nm)	R^2
F_x	0.45	0.91	0.67	0.80
F_z	0.79	0.99	1.49	0.96
M_y	0.07	0.87	0.11	0.66

Table 3: RMS Error and R^2 for the loads predicted by each model

The greatest change in the absolute RMS error can be seen in F_z, with a 0.70 N increase. The R^2 has a corresponding decline from 0.99 to 0.96 for this case; this is still a high confidence in the fit. This increase in error for the summation method is seen in Figure 4 to be due to a reduced sensitivity to AOA at higher wind speeds. Decomposing F_z into the wind-only and wind-rotor terms in Figure 5 shows that the lack of AOA variability for the summation method is due to the wind-only term. This is clear with a wind speed of 10 m s^{-1}, where the summation curve is flatter than the whole-aircraft model. In contrast, the wind-rotor component of summation aligns with the whole-aircraft model. Note that the rotor-only term is constant for all AOA and wind speeds for both models, hence is not shown in Figure 5.

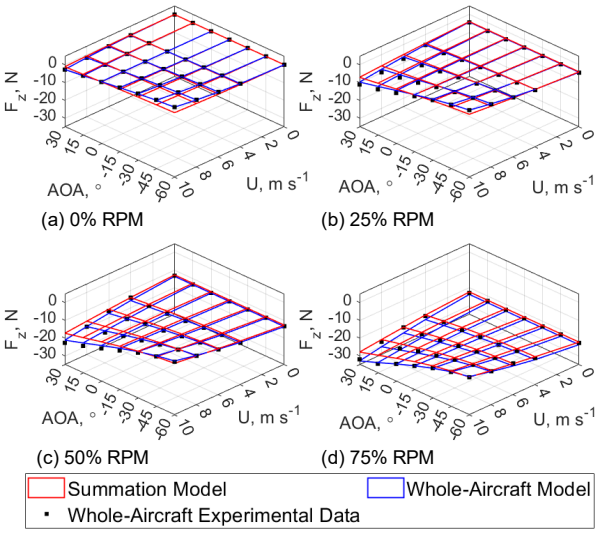


Figure 4: F_z predicted by each model

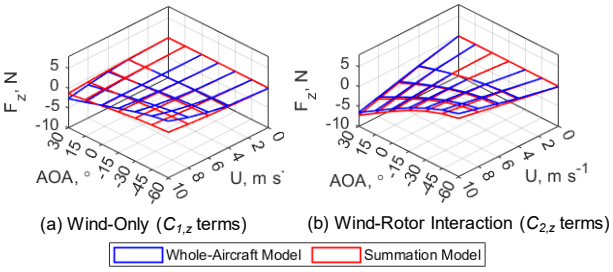


Figure 5: F_z at 75% RPM, decomposed into wind-only and wind-rotor contributions

An increase in the wind-only F_z is expected at non-zero AOA, with a positive force at negative AOA: as the aircraft tilts nose-down, the positive Z-axis rotates towards the mean wind vector, and so a greater portion of F_z is the result of airframe drag. This is seen for the whole-aircraft case, yet not the summation method in Figure 5a. This is surprising, as it would be assumed that F_z is dominated by the large UAV body at high AOA.

In contrast, the wind-rotor terms, on account of rotor sensitivity to inflow conditions, were expected to be more prone to the effects of incident wake from upstream components. However, Figure 5b shows no significant difference in the wind-rotor interaction term between the two methods. Despite these errors, F_z maintains the highest R^2, due to its relatively high force magnitude (the rotors primarily producing thrust in the aircraft Z-axis). Furthermore, the aircraft is unlikely to attain AOA exceeding ±30° in a hover condition. Hence, the differences at high AOA can be mitigated.

In comparison, M_y and F_x have relatively low R^2. This is because of their relatively low mean values, as shown by Figures 6 and 7, respectively. The difference in the whole-method and summation can again be seen to be due to differences in AOA dependence. For M_y, the overall trend is still followed, with a negative AOA producing a negative moment and an increase in both wind and rotor speed increasing the moment.

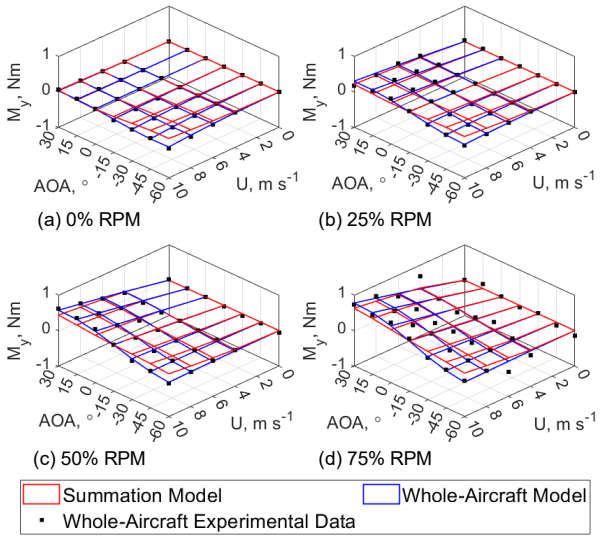


Figure 6: M_y predicted by each model

F_x, on the other hand, shows a difference in the form of the models around 0° AOA. There is a decrease in F_x magnitude at this AOA for the summation method, increasing to a more negative F_x at non-zero AOA until 45° is reached, at which point the magnitude decreases again. In contrast, the whole-aircraft model shows a gradually more positive F_x for any deviation from 0° AOA. Interestingly, the wind tunnel data shows both

behaviours, with the 0° local minimum more evident at higher rotor speeds, suggesting neither model is particularly favourable in this case.

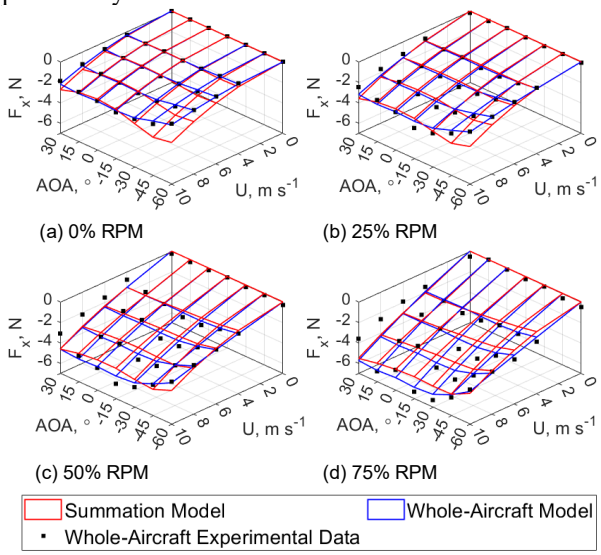


Figure 7: F_x predicted by each model

Figure 8 shows that not only do the models differ in form, but so does the experimental data for F_x from the wind tunnel, seen as black dots in each sub-plot. For the body-only test, no propellers were installed, while the whole-aircraft C_1 terms the propellers were installed, but unpowered. The propellers were free to windmill, representing the only physical difference in the two experimental data sets. A minimum F_x magnitude at 0° is expected, as this is when the aircraft is presenting minimal frontal area, hence minimum drag is expected. Deviating from this increases the projected area, and so drag, until the rotation of the X-axis away from the wind vector reduces the influence of drag on F_x. This is the behaviour seen in the body-only case, both in the model and the experimental data, but not so for the whole-aircraft. While this difference results in the whole-aircraft model being more representative at low rotor speeds, it is unlikely that the aircraft will operate with the rotors unpowered in hover. At higher rotor speeds, neither method captures the exact form of F_x,

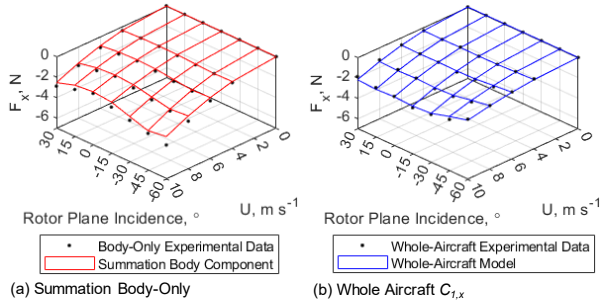


Figure 8: Fit of (a) the body-only F_x and (b) the whole-aircraft wind-only component F_x to wind tunnel data

Overall, the summation model captures the trends of F_z and M_y, but displays increasing error at high AOA. This suggests that the summation method may not be viable with higher wind speeds, where greater AOA are required, but may be appropriate at low wind speeds.

3.2 Station-Keeping Simulation

Using the coefficient fits obtained from the wind tunnel testing, Figure 9 shows the RMS position error, in millimetres, for both simulation with the whole-aircraft and summation methods. Additionally, the wind tunnel flight test data can be seen for mean wind speeds of 0, 5.6 and 12.8 m s^{-1}. Both models underestimate the position error in the no-wind condition, by an order of magnitude, with little difference between the models. Indeed, the wind tunnel flight test unexpectedly shows a higher position error for the no-wind case than with a wind speed of 5.6 m s^{-1}. This is likely due to the aircraft operating in its recirculated wake: as the wind tunnel has a closed-wall test section, in still air the wake from the rotors deflects off the floor and walls and back on to the aircraft. As the mean wind speed increases, this unsteady wake is convected downstream, reducing its impact on the aircraft. As the simulation does not suffer from wall effects, the expected deviation is much lower than the wind tunnel test without wind. Hence, this difference is likely due to experimental conditions.

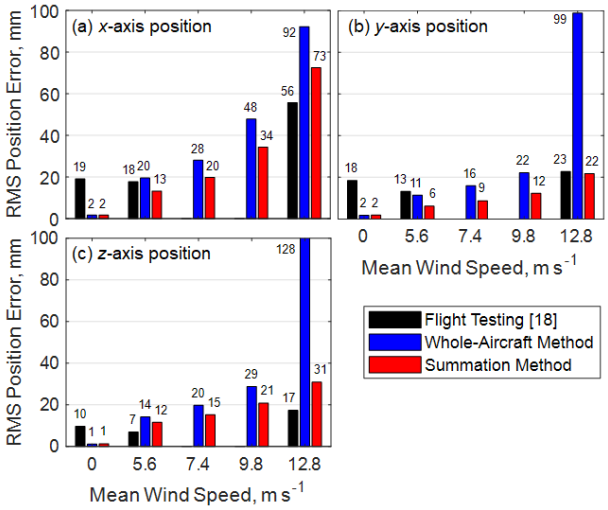


Figure 9: RMS Position Error for each model

Increasing the wind speed produces an increase in the position error predicted by both simulation methods. This is to be expected, due to the increased onset turbulent wind disturbance, and aligns with the experimental data. However, the predicted error for the simulations, particularly the whole-aircraft method, exceeds that of the flight test, most notably at 12.8 m s^{-1}. This is likely due to two critical assumptions of the simulation: quasi-steady response and uniform flow across the entire

http://www.imavs.org/

http://www.imavs.org/

airframe. In reality, the flow is not uniform across the airframe due to the irregular, chaotic nature of turbulence, in addition to flow deformation due to the aircraft. This, in turn, reduces the net variation in load on the aircraft [19].

While these inaccuracies are inherent to the simulation, the greater position error for the whole-aircraft model highlights a difference in modelling methods. Interestingly, this places the summation method more in alignment with the experimental data. Although interference effects are expected to be better handled through the whole-aircraft model, the latter may also produce excessively-variable rotor speeds to produce a desired force. This is because the local AOA at each rotor was not considered during the coefficient fits. As such, the whole-aircraft model cannot accurately predict the contribution from each rotor when they are no longer run at a uniform speed. In contrast, the summation model is based on the individual rotor AOA.

To investigate this further, Figure 10 shows a sample of the predicted aircraft forces for $U = 12.8$ m s^{-1}. It should be noted that the incident flow is identical at a given timestep for the two cases. The signal for the whole-aircraft method (on left) is considerably noisier than that of the summation method (on right).

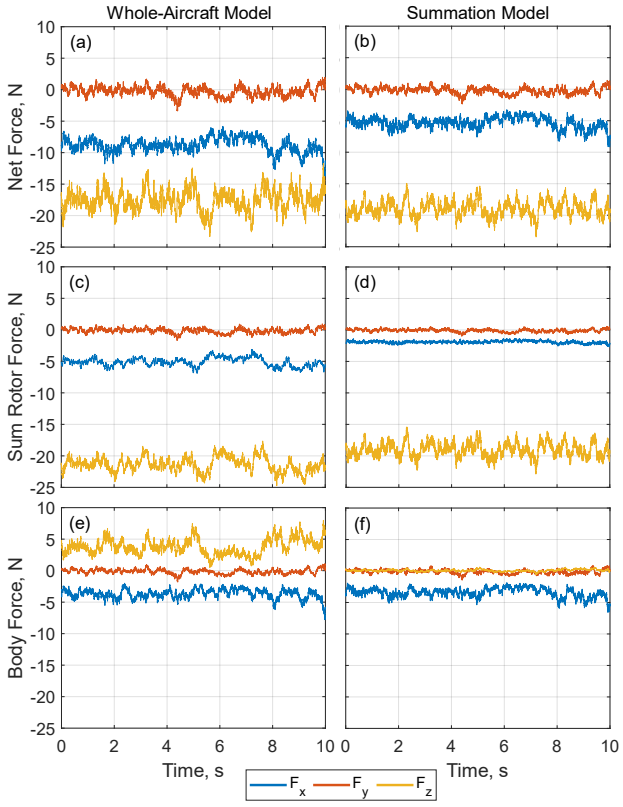


Figure 10: Simulation force estimates for (left) Whole-Aircraft Model and (right) Summation Model when U = 12.8 m s-1 for (a-b) the net aircraft, (c-d) the total rotor force and (e-f) the body contribution

The greater variation in the whole-aircraft model is to be expected, given the greater AOA sensitivity seen in the former combined with the quasi-steady assumption. The body-only force is the primary indicator of this, as the rotor-only and net aircraft estimates are altered by the commanded inputs. Not only does F_z for the body show considerably more variation, but also that its magnitude is greater. In the summation method, F_z is near-zero throughout the period shown, while the whole aircraft model produces +5 N. While this may be due to the increased AOA sensitivity, a higher drag estimate may also result in a greater nose-down attitude. More thrust needed to offset the higher drag, thus increasing F_z as the axis rotates towards the drag vector. As such force components cannot be measured from the free-flight test, this cannot be directly validated against experiments.

However, the pitch angle may be used to gauge drag estimates. As the horizontal thrust capability was not employed during the free flight tests in the wind tunnel, the aircraft must tilt to resist drag. Figure 11 shows the pitch angle of the aircraft in flight and in the simulations.

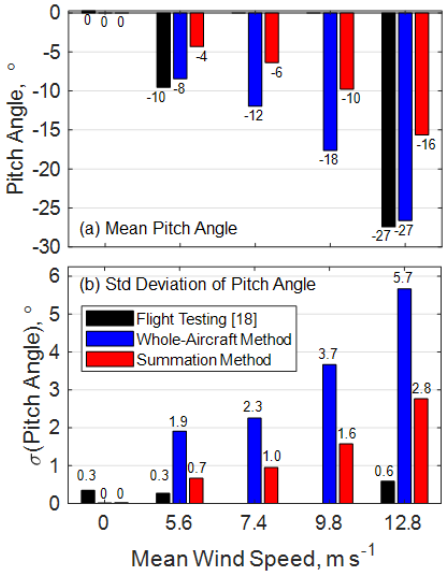


Figure 11: Mean and standard deviation of the Pitch Angle for each model compared with wind tunnel data

With its better fit to the steady-state aerodynamic loads, particularly M_y and F_x in this case, the whole-aircraft model produces a similar mean pitch angle to the free-flight data. The summation model, which was less sensitive to AoA and predicted lower forces, produces a lesser mean pitch. This suggests that the force estimates of the whole-aircraft model are, on average, nearer the expected. This underestimation of the pitch angle for the summation method is important as it could result in an aircraft being built with insufficient pitch authority. Furthermore, in roles requiring precision, such as object interaction or photography, the increased pitch angle experienced in

flight compared to the model would result in greater positional error for the end effector.

The transient pitch angle ranges for both models are exacerbated by the quasi-steady assumption, and greatly exceed the wind tunnel range. With its greater AOA sensitivity, the whole-aircraft model experiences a greater pitch angle range than its summation counterpart. This excessive motion may lead to excessively aggressive tuning during controller development.

4 DISCUSSION

The major limitation of this work is the reliance on data for a single aircraft. Although this means that the trends cannot be quantified for all configurations, it does nonetheless provide an insight into the issues with summation. For example, all axes show an increase in error when using the summation method to generate force estimates, compared to a model based on the whole aircraft. This is inevitable, as the former has no means of accounting for component interaction. As such, aircraft with considerable rotor-airframe overlap can be expected to deviate further from the true state when using summation methods. The question, however, is the viability of summation for initial aircraft performance estimates, rather than high-fidelity simulation.

The lack of AOA sensitivity, in particular, could lead to issues when using the summation method. This is mainly through the risk of underestimating pitch authority requirements in higher wind speeds. Caprace et al. [10] noted that rotor-frame interaction contributed significantly to the pitch moment. As such, summation is more suited for low-speed environments, with the whole-aircraft model more appropriate for higher flow speeds. However, this would require prototype tests or high-fidelity simulation data to perform the necessary fits. However, such accuracy comes at a cost: Caprace et al. [10] used 420 CPU hours per rear rotor revolution for their vortex method to account for rotor interaction. In contrast, the simulation employed here required fewer than 150 CPU seconds for the full 700 second flight. Although this low solving cost applies to both the whole-aircraft and summation methods, the input data for the latter is much more accessible through data sheets and low-order models. As such, at low wind speeds, summation is a valuable tool for rapid development.

A limitation which may not be as avoidable at low wind speeds is overlapped rotors. Although canted rotors do interact with each other, this is not to the extent of fully-coaxial or directly overlapping designs. This means that novel configurations, such as that proposed by Al-Zubaidi and Stol [6], would not lend themselves to a simple summation, as the rotors are always, and significantly, interacting in all phases of flight. At minimum, a thrust reduction model would be needed for the coaxial pair. It may be suitable to use a simple BEMT-based thrust deficit, but this would require testing of such an aircraft for validation.

4. CONCLUSIONS

The aerodynamic loads acting on an octorotor have been assessed via a modified empirical aerodynamic model, estimating the loads acting on a multirotor aircraft as the sum of the loads on individual components. Despite neglecting interaction effects, for a quasi-planar airframe configuration, this model produces steady force estimates comparable to that of the whole-aircraft case, albeit with increasing error at high wind speeds. Flight simulations, however, showed that the unsteady response is much less than for a whole-aircraft equivalent, with the mean pitch angle also differing from the wind tunnel expectations. Despite this, the summation method provides a reasonable estimate of the aerodynamic loads at wind speeds below 10 m s^{-1} and may prove valuable for initial development of an aircraft against the criteria of hover performance.

ACKNOWLEDGEMENTS

The research reported in this article was conducted as part of “Enabling unmanned aerial vehicles (drones) to use tools in complex dynamic environments UOCX2104”, which is funded by the New Zealand Ministry of Business, Innovation and Employment.

REFERENCES

[1] J. Walker, "Industrial Uses of Drones—5 Current Business Applications," ed: TechEmergence, 2017.

[2] R. Austin, *Unmanned aircraft systems: UAVS design, development, and deployment*, 1 ed. (no. Book, Whole). West Sussex, UK: John Wiley & Sons, 2010.

[3] ICAO, "ICAO Cir 328 Unmanned Aerial Systems (UAS)," in "Circular," Montreal, Canada, 2011. [Online]. Available: https://www.icao.int/meetings/uas/documents/circular%20328_en.pdf

[4] P. J. Richards, K. A. Stol, J. Basra, T. F. Walker, J. X. J. Bannwarth, and Z. J. Chen, "Station Keeping of Multi-Rotor UAVs in Windy Conditions," presented at the 19th Australasian Wind Engineering Society Workshop, Torquay, Vic, Australia, April 4-6, 2018.

[5] S. Delbecq, M. Budinger, A. Ochotorena, A. Reysset, and F. Defaÿ, "Efficient sizing and optimization of multirotor drones based on scaling laws and similarity models," *Aerospace Science and Technology,* vol. 102, p. 105873, 2020.

[6] S. Al-Zubaidi and K. A. Stol, "Preliminary design optimisation of a novel fixed-tilt heterogeneous UAV for horizontal agility," in *2022 International*

http://www.imavs.org/

http://www.imavs.org/

Conference on Unmanned Aircraft Systems (ICUAS), 2022: IEEE, pp. 1489-1496.

[7] H. Zhu, H. Nie, L. Zhang, X. Wei, and M. Zhang, "Design and assessment of octocopter drones with improved aerodynamic efficiency and performance," *Aerospace Science and Technology,* vol. 106, p. 106206, 2020.

[8] M. Brazinskas, S. D. Prior, and J. P. Scanlan, "An empirical study of overlapping rotor interference for a small unmanned aircraft propulsion system," *Aerospace,* vol. 3, no. 4, p. 32, 2016.

[9] B. Theys, G. Dimitriadis, P. Hendrick, and J. De Schutter, "Influence of propeller configuration on propulsion system efficiency of multi-rotor Unmanned Aerial Vehicles," in *2016 international conference on unmanned aircraft systems (ICUAS)*, 2016: IEEE, pp. 195-201.

[10] D.-G. Caprace, A. Ning, P. Chatelain, and G. Winckelmans, "Effects of rotor-airframe interaction on the aeromechanics and wake of a quadcopter in forward flight," *Aerospace Science and Technology,* vol. 130, p. 107899, 2022.

[11] S. Watkins, J. Milbank, B. J. Loxton, and W. H. Melbourne, "Atmospheric winds and their implications for Micro Air Vehicles," *AIAA Journal,* vol. 44, no. 11, pp. 2591-2600, 2006, doi: https://doi.org/10.2514/1.22670.

[12] M. Cerny and C. Breitsamter, "Investigation of small-scale propellers under non-axial inflow conditions," *Aerospace Science and Technology,* vol. 106, p. 106048, 2020.

[13] C. Rodríguez de Cos and J. Á. Acosta, "Explicit aerodynamic model characterization of a multirotor unmanned aerial vehicle in quasi-steady flight," *Journal of Computational and Nonlinear Dynamics,* vol. 15, no. 8, p. 081005, 2020.

[14] M. C. Ruiz, M. Scanavino, D. D'Ambrosio, G. Guglieri, and A. Vilardi, "Experimental and numerical analysis of hovering multicopter performance in low-Reynolds number conditions," *Aerospace Science and Technology,* vol. 128, p. 107777, 2022.

[15] Z. Chen, K. Stol, and P. Richards, "Preliminary design of multirotor UAVs with tilted-rotors for improved disturbance rejection capability," *Aerospace Science and Technology,* vol. 92, pp. 635-643, 2019.

[16] Z. J. Chen, J. X. Bannwarth, K. A. Stol, and P. J. Richards, "Analysis of a Multirotor UAV with Tilted-rotors for the Purposes of Disturbance Rejection," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2018: IEEE, pp. 864-873.

[17] J. X. J. Bannwarth, Z. J. Chen, K. A. Stol, B. A. MacDonald, and P. J. Richards, "Aerodynamic Force Modeling of Multirotor Unmanned Aerial Vehicles," *AIAA Journal,* vol. 57, no. 3, pp. 1250-1259, 2019, doi: 10.2514/1.J057165.

[18] J. X. J. Bannwarth, "Aerodynamic Modelling and Wind Disturbance Rejection of Multirotor Unmanned Aerial Vehicles," PhD, Department of Mechanical Engineering, The University of Auckland, Auckland, New Zealand, 2022.

[19] I. Andreu-Angulo, H. Babinsky, H. Biler, G. Sedky, and A. R. Jones, "Effect of Transverse Gust Velocity Profiles," *AIAA Journal,* vol. 59, no. 1, pp. 1-11, 2020, doi: https://doi.org/10.2514/1.j059678.

APPENDIX A: COEFFICIENT FITS

Whole-Aircraft
$C_{1,x} = -0.2003\cos(\alpha) + 0.0060\cos(3\alpha) + 0.0102\cos(7\alpha)$
$C_{2,x} = -0.0396\left(1 - e^{-0.1584\lambda}\right)\cos(\alpha) + 0.0086\left(1 - e^{-0.0565\lambda}\right)\cos(3\alpha)$
$C_{1,z} = -0.4326\sin(\alpha) - 0.0055$
$C_{2,z} = -0.0815\sin(\alpha) + 0.4583\left(1 - e^{-9.31\times10^{-4}\lambda}\right)\sin(3\alpha) - 0.0175$
$C_{3,z} = -0.0152$
$C_{M1,y} = -0.0150\cos(\alpha) + 0.0126\cos(3\alpha) + 0.0271\sin(2\alpha)$
$C_{M2,y} = 0.0549\left(1 - e^{-0.5307\lambda}\right)\cos(\alpha) + 0.2729\left(1 - e^{-0.1174\lambda}\right)\sin(2\alpha)$
Rotor-Only
$C_{1,x} = -0.0076\cos(\alpha) - 0.0055\cos(3\alpha) + 0.0055\cos(7\alpha)$
$C_{2,x} = -0.0330\left(1 - e^{-0.0349\lambda}\right)\cos(\alpha) - 0.0031\left(1 - e^{-0.0735\lambda}\right)\cos(3\alpha)$
$C_{1,z} = -0.1124\sin(\alpha) + 0.0025$
$C_{2,z} = -0.1039\sin(\alpha) + 2.192\left(1 - e^{-5.18\times10^{-4}\lambda}\right)\sin(3\alpha) - 0.0308$
$C_{3,z} = -0.0149$
$C_{M1,y} = 0.0232\cos(\alpha) - 0.0140\cos(3\alpha) + 0.0236\sin(2\alpha)$
$C_{M2,y} = 0.0652\left(1 - e^{-0.0632\lambda}\right)\cos(\alpha) + 0.0630\left(1 - e^{-0.0278\lambda}\right)\sin(2\alpha)$
Body-Only
$C_{1,x} = -0.2524\cos(\alpha) + 0.1095\cos(3\alpha) - 0.0062\cos(7\alpha)$
$C_{1,z} = -0.0845\sin(\alpha) - 0.0192$
$C_{M1,y} = 0.0103\cos(\alpha) - 0.0227\cos(3\alpha) - 0.0058\sin(2\alpha)$

Table 4: Coefficient Fits for Each Model

Swarm based airborne wind measurement

S. Kean*, M. Marino, S. Watkins, A. Mohamed
RMIT University, Melbourne, Australia

ABSTRACT

This paper presents the design and development of a swarm of four wind sensing MAVs equipped with custom four-hole pressure probes to record airspeed and direction at up to 1000Hz. A comparison between a TFI Cobra Probe and the drone system is made during a hovering experiment. Following this an atmospheric boundary layer is measured using the four drones simultaneously. Finally, a preliminary attempt at mapping the wind flow around a shipping container is made.

http://www.imavs.org/

1 INTRODUCTION

There is a growing desire for a deeper understanding of the local flow conditions within the lower regions of the atmospheric boundary layer. This stems from a number of applications including localized weather predictions, particulate dispersion, wind energy extraction, pedestrian comfort and advanced air mobility (AAM) concepts such as air taxis and micro air vehicles (MAV) for package delivery. Many of these emerging applications will take place in urban environments, which are dominated by mechanical mixing due to high surface roughness and the presence of buildings. This mechanical mixing produces high levels of turbulence that occur at a wide range of spatial and temporal scales, producing a complex spectrum of eddies[1]. Reproducing the full spectrum in a wind tunnel or CFD model is challenging, time consuming and requires accurate modeling of the location and upwind fetch. An alternative is to directly measure the flow field via full-scale experiments. This is typically done using combinations of statically mounted anemometers and long-range sampling devices like SODARs and LIDARs. These techniques are often limited by the placement of devices and the cost of installing masts and other structures to reach a desired location and are inherently expensive and prohibitive. Long range measurements devices are also often unsuitable for studying small scale turbulence as they average over a measurement volume[2]. For these reasons, it is expected that MAV-based measurements could provide a cheap, flexible and configurable solution to collecting short time scale wind measurements in previously inaccessible locations.

A wealth of knowledge already exists regarding the wind loading of structures, however much of this previous work examines wind flows at much longer spatial and temporal scales than are expected to be relevant for these emerging applications. Wind engineering standards vary from country to country, but generally use time averages of 10 minutes in meteorology [3] up to 30 minutes or more as required by the applications for assessing structural loads[4]. For the case of predicting wind energy availability and the placement of micro turbines, Stathopoulos et al. highlight direct measurements as the most dependable method, despite the inherent costs required[5]. However, this is usually not possible during the design phase, as no building exists at the site to mount anemometers on. Hence the use of CFD and wind tunnel modeling as a means to predict the flow environment after construction[6]. Both of these solutions, however, require reliable knowledge of the local inflow conditions at the site to achieve realistic replication. Inflow conditions are usually estimated from standards based from the terrain classification at the measurement location and its fetch[6, 1]. MAVs have been shown to be an effective tool for measuring these inflow conditions by collecting ABL profiles at a variety of terrain conditions[7, 8, 9] including sub-urban and urban environments[10]. In most of the studies these profiles were validated against measurements taken on a static mast.

MAVs have seen increasing use for the measurement of atmospheric phenomena, utilizing both fixed wing and multirotor configurations depending on the studies performed[11]. The flexibility of an airborne platform provides unique benefits, particularly for difficult to reach locations such as around buildings and wind turbines. In the last 5 years there has been a large focus on the use of indirect measurement techniques, where the wind vector is inferred from a kinematic model of the vehicle[12, 13, 14, 15]. This is a cost-effective solution as nearly any off-the-shelf drone can be utilized, provided the kinematics of the vehicle can be accurately modeled. Indirect techniques, however, are generally not suitable for the study of turbulent properties as they spatially average the volume that the drone occupies and rely on the response of the drones control system which varies greatly from platform to platform. Alternatively, direct techniques have also been explored[10, 16, 17], utilizing anemometers mounted on the multirotor to directly sample the flow field. Depending on the anemometer chosen, this can enable the measurement of turbulent properties. A key concern often raised about such systems is ensuring the aircraft does not interfere with the measurements. This is done either by careful placement of the device, post processing corrections to remove influences or a combination of both. Barbieri et. al performed an intercomparison of a collection of meteorology MAVs in a series of field trials and compared the results against a static

*Email address: s3605238@student.rmit.edu.au

http://www.imavs.org/

mast. In the case of flow sensing, they found that all the multirotors showed good agreement with the tower based measurements, often outperforming their fixed wing counterparts due to the ability to fly stationary relative to the reference instrument[11].

MAVs can also be coordinated in a swarm to collect simultaneous data, allowing for more complex analysis to be performed. Pu et al.[18] Utilized simultaneous measurements with two MAVs using sonic anemometers to develop a gradient map of wind speeds and turbulent intensities in an urban environment. The first MAV was used to measure the upstream flow during the experiment at a fixed reference height, whist the second recorded the flow between buildings. Larger swarms have also been demonstrated such as Wetz et al.[9] comparing indirect wind measurements from 10 MAVs against mast mounted sonic anemometers and lidars, showing that flow structures in time and space could be resolved and compared to static references.

2 OBJECTIVES

To provide a brief overview of the measurement, flight and post processing system, then demonstrate the capability of a swarm of four flying anemometers to provide high frequency wind data. Three experiments will be conducted. A hovering comparison against a TFI Cobra Probe, recording of an atmospheric boundary layer profile and 3D visualization of the wind flow around a shipping container.

3 SYSTEM

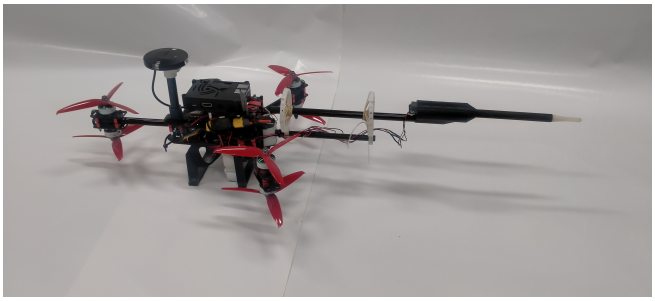


Figure 1: One of four MAVs with measurement system.

The MAVs were designed to provide a stable measurement platform in turbulent and gusty conditions, as well as keep the probe as far away from the rotors as possible. For this reason, a Y6 frame was chosen, as the asymmetry lends itself to mounting the pressure probe on a damped mast (see Figure 1). It is also convenient to use smaller props as it means the probe tip can be closer to the CG of the drone. Prudden et al. previously investigated the interaction between air speed and direction for a similar MAV configuration and determined that at a distance of at least 2.5 rotor diameters air speed was not measurably effected and flow deflection was minimized[19]. As such the probe tip is mounted 3 rotor diameters upstream in the above design. Wind tunnel experiments are planned to validate and potentially correct any angle changes induced by the props.

The measurement system is a custom 4-hole pressure probe designed to be mounted on an MAV. The probe can measure airspeed and direction at 1000Hz and was designed after the TFI Cobra Probe. The probe tip was 3D printed in resin with a tip diameter of approximately 1cm. A batch of the printed probe heads was calibrated and intercompared in a wind tunnel and were found to have no measurable variation between the prints. This means that if the probe head is damaged it can be replaced without needing to re-calibrate in the wind tunnel.

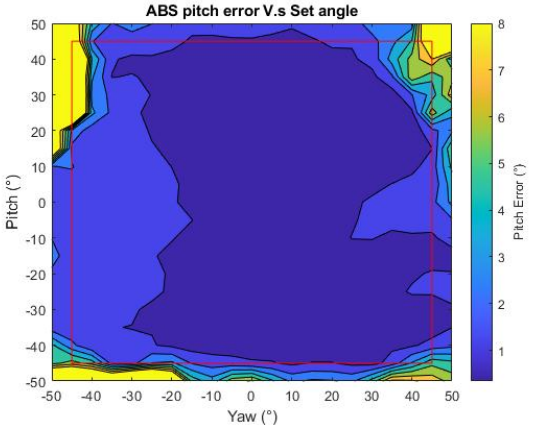


Figure 2: ABS Pitch error

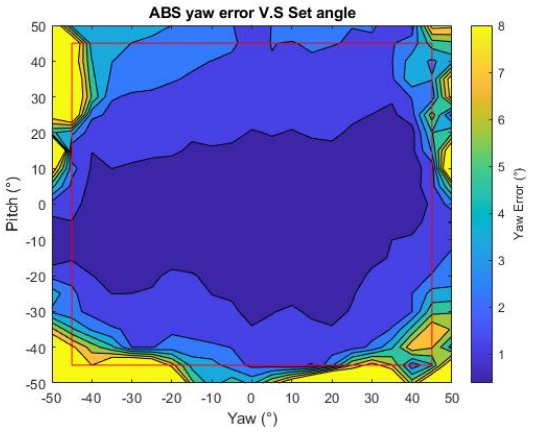


Figure 3: ABS Yaw error

The results of the calibration process showed that the maximum error for pitch, yaw and speed was ±2°, ±2° and ± 5%. The plots in Figure 2 depict this, where the x and y axis correspond to the set pitch and yaw angle of the probe, and the color represents the difference between the measured pitch (Figure 2), yaw (Figure 3) and speed (Figure 4). Six-

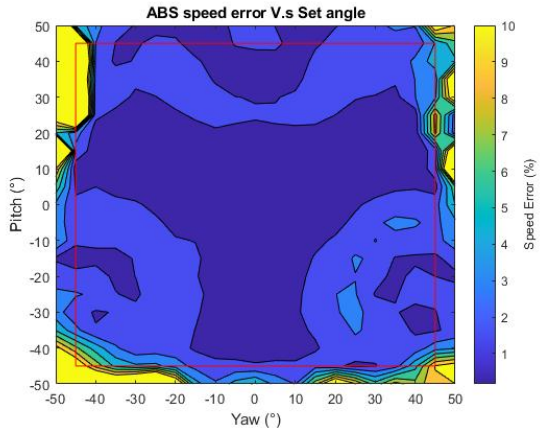
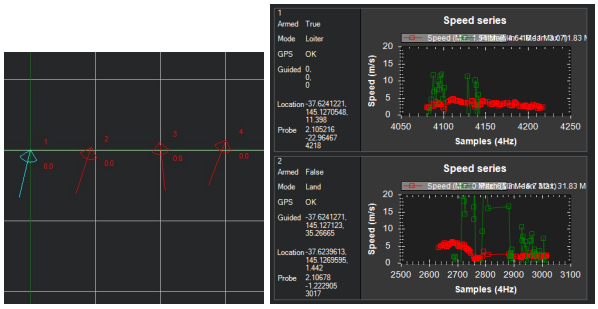


Figure 4: ABS Speed error

teen calibration functions were generated for this data set, using four probe heads and four speeds, nominally 5, 10 and 15 m/s. These can be compared as the calibration surfaces are non-dimensional and Reynolds number invariant[20]. It is worth noting that these values represent the largest errors observed at the extremities of the 45° cone of acceptance which is overlayed in Figure 2 using the red box. Within the first 25 degrees of the origin in both pitch and yaw axis, the error is less than ±1°.

To correctly represent the wind vector in an Earth-centric coordinate system, the motion and attitude of the drone needs to be corrected for. This is done using the flight log from the drone, and using a sequence of transformations based on the equations in Prudden et al[16]. These equations are used to subtract any airspeed induced by the motion of the drone both linear and rotational. This means that the drones can be in motion when recording wind data, although the most accurate results are achieved when close to constant motion is maintained. Once this is complete, the data is low passed and down sampled to 120Hz. This is conveniently below the blade pass frequency, one of the largest sources of mechanical noise.

Another challenge for this is system is ensuring temporal coherence between each of the separate MAVs. The GPS UTC time provided in the flight log is not very reliable, showing significant drift between separate drones. By interfacing a raspberry Pi with the flight controller, the electrical trigger of the probe can be recorded with much higher accuracy. The probe is then interfaced with the Pi over USB to record the pressure measurements. The Pi listens to the telemetry port on the flight controller and when a UTC boot time is recorded from the GPS it is compared with Pi's own UTC time. The difference is used to align the two flight logs under the assumption that the electrical signal is perceived simultaneously by the Raspberry Pi and the probe. This also enables the Pi to publish probe sensor data into the MAVLINK telemetry stream. This data is transmitted via the telemetry radio to the ground control station for use during experiments. A screenshot of the swarm control interface is shown in Figure 5.



(a) (b)

Figure 5: (a) GCS Drone formation view (b) GCS drone status and probe view

The swarm control system is a modified version of the beta swarm feature included in Mission Planner- the open source ground control station(GCS). The modifications include a critical bug fix that caused violent oscillations in the vehicles yaw axis, as well as a rewrite of the interface to enable it to be more easily utilized for atmospheric measurement purposes. Conceptually, the swarm system is a formation flight, where each follower vehicle receives guided mode commands from the GCS. The interface enables the vertical and horizontal offsets of each vehicle to be specified relative to the flight leader. It is then up to the autopilot of each vehicle to achieve these waypoints and maintain the correct locations. Although this is not an intelligent or adaptive control system, its more than sufficient to support the atmospheric measurements that this research requires. Having a centralized control system also enables the mission to be easily modified during a flight, something that is challenging with preprogrammed waypoint controls. The final advantage of this system, is that the swarm control system is only concerned with high level vehicle commands, leaving flight operations to the onboard flight controller, meaning that the control system is agnostic of the vehicle hardware.

4 EXPERIMENTS

4.1 Hovering comparison

To validate the whole system in combination, a flight experiment was performed where an MAV was hovered next to a cobra probe on a 3m mast for approximately two minutes. The drone was positioned with its tip 1m to the right of the cobra probe and facing in approximately the same direction. Both the cobra probe and the drone were recording at 1000Hz. Figure 6 shows good agreement for the low frequency components and the average speed over the entire sample was 4.95m/s for the cobra probe and 5.13m/s for the MAV. Turbulent intensity is a non-dimensional measure of

http://www.imavs.org/

the fluctuating component of a velocity time series. It can be calculated along each component axis as shown in (1), or as a total as shown in(2).

$$Ti_u = \frac{\sigma_u}{\overline{V}}, Ti_v = \frac{\sigma_v}{\overline{V}}, Ti_w = \frac{\sigma_w}{\overline{V}} \quad (1)$$

$$Ti_{total} = \frac{\sqrt{\sigma_u^2 + \sigma_v^2 + \sigma_w^2}}{\overline{V}} \quad (2)$$

The turbulence intensities, shown in Table 1, also showing good agreement. Some differences are to be expected between the two measurements, especially in the short time scale. Such velocity fluctuations can occur across very small spans and have been previously measured in [21].

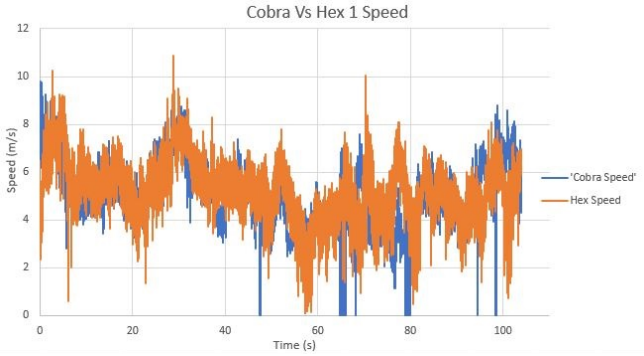


Figure 6: Wind speed comparison between static cobra probe and flying MAV probe

	Avg. Speed(m/s)	Total	Ti_u	Ti_v	Ti_w
MAV	5.13	0.24	0.24	0.29	0.2
Cobra	4.95	0.22	0.26	0.24	0.13

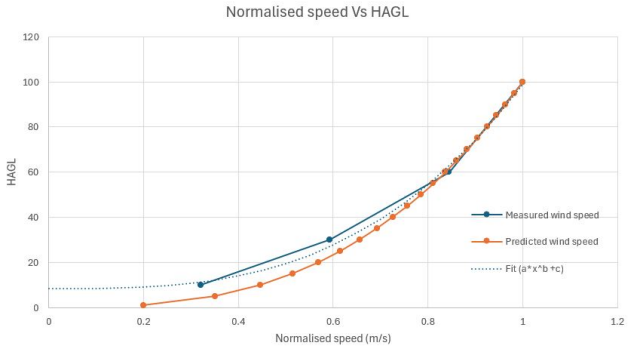
Table 1: Comparison between Average speeds and turbulence intensities.

4.2 *Atmospheric boundary layer profile*

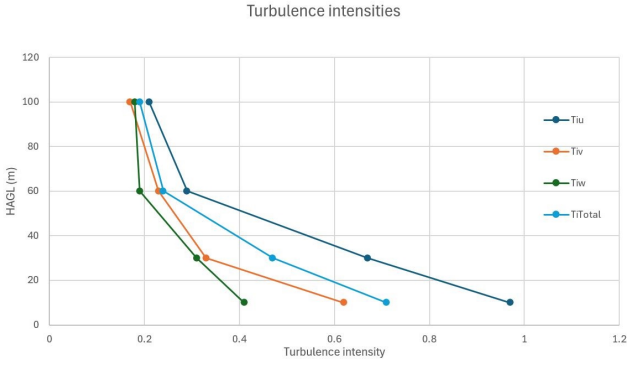
Atmospheric boundary layer profiles provide information about how the wind interacts with the local environment. As the wind flows across the surface of the planet, it interacts with any buildings or objects in its path. This creates a decaying spectrum of turbulence close to the ground. As a result, wind speed generally increases while turbulence intensity decreases with altitude exponentially. The power law is one model of wind profile used within the lower ABL and is shown in (3).

$$u(z) = u(z_0) * (\frac{z}{z_0})^a \quad (3)$$

Assuming neutral stability conditions, the exponent a is entirely dependent on the roughness of the local terrain, and therefore can be calculated by directly measuring the wind at multiple heights. Standards provide typical ranges for the curve exponent based on the roughness classifications of different terrains. Forests and sub-urban areas, for example, are listed by Emeis et al. as being 0.28-0.4 [22]. The experiment was performed at Greensborough Model Aircraft Club, in a flat open field. Roughly 100m upwind of the measurement site is a row of trees, approximately 20m in height, followed by 2km of sparsely wooded parkland. The landscape quickly develops into an urban sprawl that continues towards Melbourne. To measure the wind speeds, the drones were coordinated as a vertical stack at 10m, 30m, 60m and 100m above ground level. The wind speed at each height was averaged and the results are shown in Figure 7 (a). The blue curve represents the 4 simultaneously measured points, and the dotted blue line is a best fit to the power law. The orange curve is a theoretical power law with an exponent value of 0.35 for comparison. This value was chosen as the 'average' for this terrain category.



(a)



(b)

Figure 7: (a) Wind speed v.s HAGL. (b) Turbulent intensity v.s HAGL

Fitting a curve to the measured data, an exponent of 0.33 is calculated. This value is a surprisingly good match to the theoretical model given the simplifications made. The fitted curve also shows that the y-intercept(displacement height), the height at which zero wind speed could be expected, to be

http://www.imavs.org/

http://www.imavs.org/

at around 8m. This is reasonable given that the upwind trees effectively created a wind shadow at the measurement site. It's worth noting that no displacement height was used for the theoretical power curve, as the equations generally don't include it and so typical values are not available. This is not the case for the logarithmic wind model in which a displacement height is specified, and typical values can be found in the ESDU data sheets[4]. Although inter comparing wind models is a challenging task unto itself, the typical displacement heights can be compared as they simply represent a translation along the y axis. The displacement values provided for urban and fairly level wooded country is between 5 and 10m, which is a good match to the measured data. Figure 7 (b) shows the change in turbulent intensities against altitude. Close to the ground a significant decrease in the w component of turbulence intensity is seen when compared to the other components. This aligns with the theory and is caused by the interaction of eddies with ground. As the eddies cannot pass through the ground, they effectively 'smear' their energy into the other two component directions. Its expected the combination of fetch terrain changes and the close upwind trees increased the levels of turbulence close to the ground when compared to the typical theoretical model. The indication of an 8m displacement height from the measured drone data also helps to explain the high turbulence levels recorded by the drones at lower altitudes.

4.3 Container flow mapping

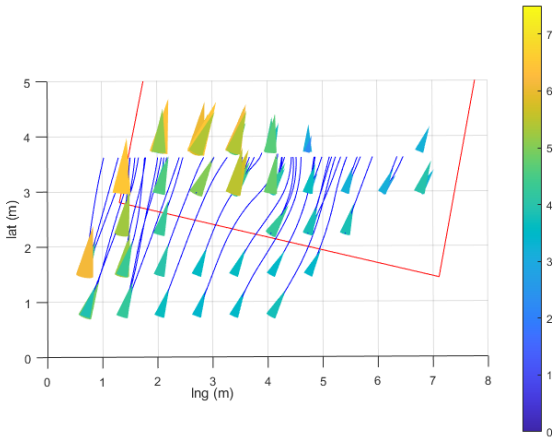
As it is ultimately intended for the drones to be used to study air flows around buildings and in urban areas, it is important to ensure that they can safely fly and record data in regions of interest. An example of this would be the shear layer that occurs near the roof of a cuboid building. A shear layer is characterized by the rapid velocity changes that occur as a result of the airflow compressing and accelerating around the object. Measuring this has particular relevance to AAM applications as any aircraft landing/taking off on a roof or vertiport has to transition through this region safely.

A single drone was manually flown around the area where the shear layer was expected to form for approximately 3 minutes and its flight path is shown by the blue line in Figure 9(a). During this time the drone faced into wind which blew from the south. The samples recorded were then processed into a 3D volume of point measurements. This was done by creating a voxel grid at the location of the flight path. The voxel grid is represented top down by the pink box in Figure 9 (a). The size of the grid cell is then used to spatially average the measurements which can then be represented using 2D slices, cones and streamlines as shown in Figure 6 and Figure 7. The grid size for below data is 0.75m.

Figure 9 (a) and Figure 9 (b) show the vertical component of wind at two different spans along the container's length. An updraft was measured along the leading edge and a down draft just behind it. It is assumed this is associated with the



(a) Satellite view of flight path



(b) Top down view of velocity magnitude

Figure 8: (a) Satellite view of container. (b) Top down view of velocity magnitude.

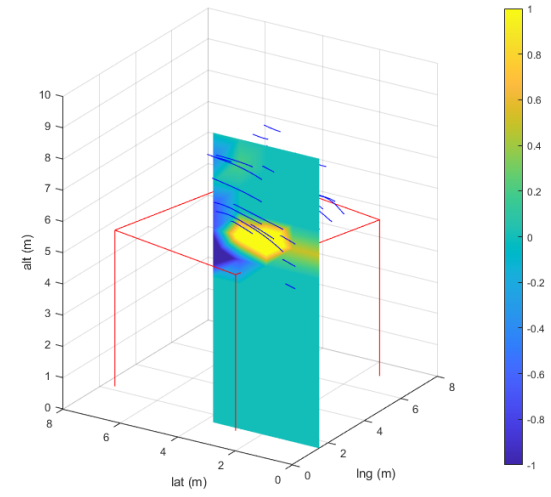
shear layer that would develop over a bluff body. Turbulence intensities were also calculated across the volume, and were shown to be higher closer to the surface of the container as shown in Figure 9 (c).

A flight path that more evenly distributes measurement points and using multiple drones would provide a clearer visualization of flow around the container, but this preliminary test shows promise. The use of AI for sparse wind field prediction is currently being researched[23, 24] and data sets such as this may in the future be completed using such predictions.

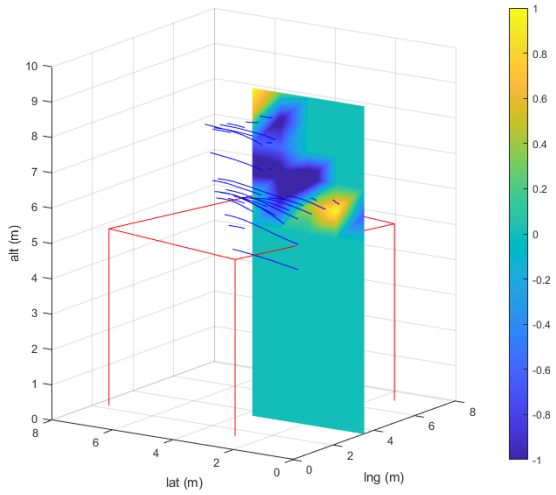
5 CONCLUSION

A swarm of four flying anemometers has been developed utilizing four-hole pressure probes to record wind speed and direction at high rates. An experiment that compared

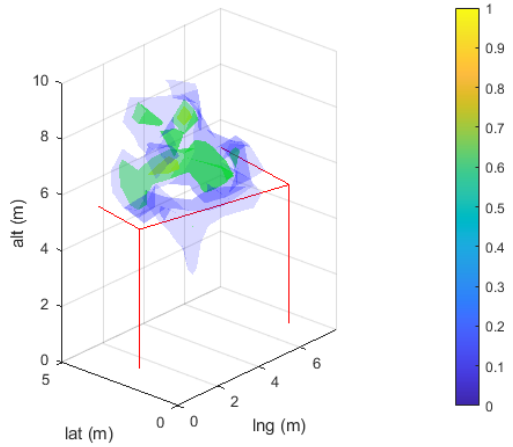
http://www.imavs.org/



(a) Vertical component of wind at x=1.5m



(b) Vertical component of wind at x=3.5m



(c) Total turbulence intensity

Figure 9: (a)Vertical component of wind at x=1m. (b)Vertical component of wind at x=3.5m. (c) Total turbulence intensity

wind speed and turbulence intensities against a cobra probe that was mounted on a mast was conducted and showed good agreement. An atmospheric boundary layer profile was recorded using all 4 drones simultaneously and the terrain exponent was calculated showing good agreement with the theoretical model. Finally, a preliminary flow mapping trial was performed around a shipping container in high winds as a scale analogy for building. An updraft was successfully measured along the leading edge of the container. In the future, full scale experiments utilising all 4 drones will be conducted on the rooftop of a multi story building. Utilising multiple drones will enable the simultaneous capture of wind data for the correlation at multiple locations. This could be used to model the role disturbance of an aircraft within this region of the flow.

REFERENCES

[1] J. C. Kaimal and J. J. Finnigan. *Atmospheric boundary layer flows their structure and measurement*. Oxford scholarship online. Oxford University Press, New York, 1st ed. edition, 1994.

[2] R. M. Hardesty. *LIDAR — Doppler*, pages 1194–1202. Academic Press, Oxford, 2003.

[3] Beurea of meteorology. About forcast models, 2024.

[4] ESDU. Characteristics of atmospheric turbulence near the ground part i: definitions and general information, 1974.

[5] Ted Stathopoulos, Hatem Alrawashdeh, Ayman Al-Quraan, Bert Blocken, Aierken Dilimulati, Marius Paraschivoiu, and Pragasen Pilay. Urban wind energy: Some views on potential and challenges. *Journal of Wind Engineering and Industrial Aerodynamics*, 179:146–157, 2018.

[6] Maryam Al Labbad, Alanna Wall, Guy L. Larose, Fidel Khouli, and Hali Barber. Experimental investigations into the effect of urban airflow characteristics on urban air mobility applications. *Journal of wind engineering and industrial aerodynamics*, 229:105126, 2022.

[7] Sara Alaoui-Sosse, Pierre Durand, Patrice Medina, Philippe Pastor, Marie Lothon, and Iuri Cernov. Ovli-ta: An unmanned aerial system for measuring profiles and turbulence in the atmospheric boundary layer. *Sensors (Basel, Switzerland)*, 19(3):581, 2019.

[8] Joachim Reuder, Marius O. Jonassen, and Haraldur Ólafsson. The small unmanned meteorological observer sumo: Recent developments and applications of a micro-uas for atmospheric boundary layer research. *Acta geophysica*, 60(5):1454–1473, 2012.

[9] Tamino Wetz, Norman Wildmann, and Frank Beyrich. Distributed wind measurements with multiple quadrotor unmanned aerial vehicles in the atmospheric boundary layer. *Atmospheric measurement techniques*, 14(5):3795–3814, 2021.

[10] Zhengnong Li, Ou Pu, Yueyue Pan, Bin Huang, Zhefei Zhao, and Honghua Wu. A study on measuring the wind field in the air using a multi-rotor uav mounted with an anemometer. *Boundary-layer meteorology*, 188(1):1–27, 2023.

[11] Lindsay Barbieri, Stephan T. Kral, Sean C. C. Bailey, Amy E. Frazier, Jamey D. Jacob, Joachim Reuder, David Brus, Phillip B. Chilson, Christopher Crick, Carrick Detweiler, Abhiram Doddi, Jack Elston, Hosein Foroutan, Javier González-Rocha, Brian R. Greene, Marcelo I. Guzman, Adam L. Houston, Ashraful Islam, Osku Kemppinen, Dale Lawrence, Elizabeth A. Pillar-Little, Shane D. Ross, Michael P. Sama, David G. Schmale, Travis J. Schuyler, Ajay Shankar, Suzanne W. Smith, Sean Waugh, Cory Dixon, Steve Borenstein, and Gijs de Boer. Intercomparison of small unmanned aircraft system (suas) measurements for atmospheric science during the lapse-rate campaign. *Sensors*, 19(9):2179, 2019.

[12] Pramod Abichandani, Deepan Lobo, Gabriel Ford, Donald Bucci, and Moshe Kam. Wind measurement and simulation techniques in multi-rotor small unmanned aerial vehicles. *IEEE access*, 8:1–1, 2020.

[13] Javier Gonzalez-Rocha, Craig A. Woolsey, Cornel Sultan, Nathan Rose, and Stephan F. J. De Wekker. *Measuring atmospheric winds from quadrotor motion*. 2017.

[14] Kilian Meier, Richard Hann, Jan Skaloud, and Arthur Garreau. Wind estimation with multirotor uavs. *Atmosphere*, 13(4):551, 2022.

[15] Meir Pachter, Nicola Ceccarelli, and Phillip Chandler. *Estimating MAV's Heading and the Wind Speed and Direction Using GPS, Inertial, and Air Speed Measurements*. 2012.

[16] S. Prudden, A. Fisher, M. Marino, A. Mohamed, S. Watkins, and G. Wild. Measuring wind with small unmanned aircraft systems. *Journal of wind engineering and industrial aerodynamics*, 176:197–210, 2018.

[17] N. Vasiljević, M. Harris, A. Tegtmeier Pedersen, G. Rolighed Thorsen, M. Pitter, J. Harris, K. Bajpai, and M. Courtney. Wind sensing with drone-mounted wind lidars: proof of concept. *Atmos. Meas. Tech.*, 13(2):521–536, 2020.

[18] Li Zhengnong Bao Terigen Chen Zheng Yang Liwei Qin Hua Li Zhen Pu Ou, Yuan Boqiu. Research on the characteristics of urban building cluster wind field based on uav wind measurement, 2023.

[19] Samuel Prudden, A Fisher, A Mohamed, and S Watkins. A flying anemometer quadrotor: Part 1. In *Proceedings of the International Micro Air Vehicle Conference (IMAV 2016), Beijing, China*, pages 17–21.

[20] I. C. Shepherd. A four hole pressure probe for fluid flow measurements in three dimensions. *Journal of fluids engineering*, 103(4):590–594, 1981.

[21] M. Thompson, S. Watkins, C. White, and J. Holmes. Span-wise wind fluctuations in open terrain as applicable to small flying craft. *Aeronautical journal*, 115(1173):693–701, 2011.

[22] Stefan Emeis. *Wind energy meteorology : atmopsheric physics for wind power generation*. Green energy technology. Springer-Verlag, Berlin, Heidelberg, 2013.

[23] Longyan Wang, Meng Chen, Zhaohui Luo, Bowen Zhang, Jian Xu, Zilu Wang, and Andy C. C. Tan. Dynamic wake field reconstruction of wind turbine through physics-informed neural network and sparse lidar data. *Energy*, 291:130401, 2024.

[24] Bingchao Zhang, Ryozo Ooka, Hideki Kikumoto, Chaoyi Hu, and KT Tim. Towards real-time prediction of velocity field around a building using generative adversarial networks based on the surface pressure from sparse sensor networks. *Journal of Wind Engineering and Industrial Aerodynamics*, 231:105243, 2022.

http://www.imavs.org/

Landing Zone Detection for MAVs using Depth Images and Vision Transformers

Victoria Eugenia Vazquez-Meza and Jose Martinez-Carranza*
Instituto Nacional de Astrofisica Optica y Electronica

http://www.imavs.org/

ABSTRACT

We present a methodology for landing zone detection in Micro Aerial Vehicles (MAVs) using Vision Transformers (ViTs). We present results on the use of ViTs due to their ability to capture spatial relations through the attention mechanism, potentially offering superior performance with fewer training examples than other Deep Neural Networks. Experiments with aerial images, from a dataset and depth images captured with a depth camera on board a drone, confirm ViT's superiority over other widely used Convolutional Network such as ResNet, particularly with limited training data. Despite the noisy depth images, captured with the depth camera, the ViT model can be used to detect landing zones with an average processing time of 11.9 ms on outdated GPU hardware.

1 INTRODUCTION

Vision Transformers (ViTs) represent a significant advancement in computer vision, leveraging the transformer architecture traditionally used in natural language processing. ViTs have demonstrated superior performance on various image classification benchmarks, often surpassing convolutional neural networks (CNNs). They handle larger datasets and more complex tasks efficiently, benefiting from the ability to learn long-range dependencies within the data. Furthermore, ViTs have been shown to learn faster with fewer examples than CNNs [1], making them suitable for tasks that require rapid prediction models. This advantage is particularly useful in scenarios, where there is limited time to collect training data and where short training times are desirable. Due to these properties, in this work, we explore the use of ViTs for landing zone detection using depth images captured with a camera on board a Micro Aerial Vehicle (MAV) that could be navigating an environment, seeking a suitable place to perform its landing.

Due to the advancements in hardware and software, today it is possible for MAVs to carry small depth cameras such as the Real-Sense or the OAK-D camera [2, 3], which provide chromatic and depth images from the scene. When depth cameras are not available, a possibility is to infer depth from chromatic images using Deep Learning [4, 5]. In both cases, we can remove the dependency on more expensive range sensors such as LiDAR or radar, which usually are bulky and expensive. Moreover, in the work of [6], it was shown that a CNN could be trained to detect more effectively landing zones using depth images as input to the network, rather than when using chromatic (RGB) images.

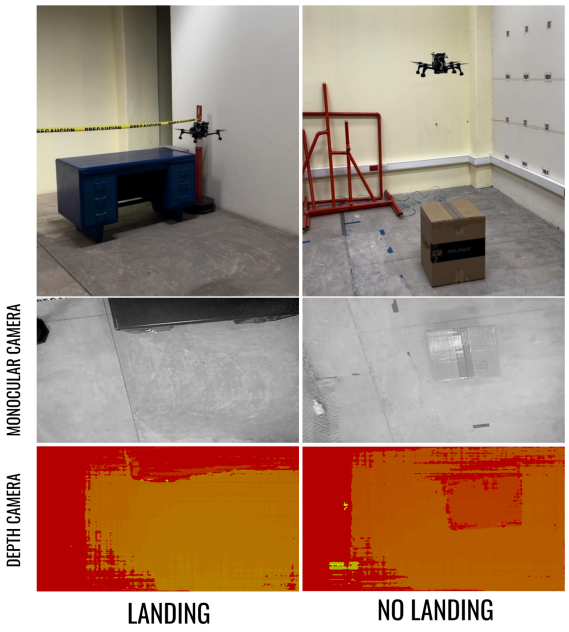


Figure 1: We leverage the attention mechanism of a Vision Transformer model to predict whether a landing zone is present in a depth image. We used the OAK-D camera onboard a MAV to obtain depth images. It is noteworthy that the chromatic image to the right barely reveals the presence of an object in the middle, which could mislead a detector based on chromatic information. In contrast, the object is more clearly visible in the depth image.

Motivated by the above, in this work, we explore the use of ViTs and compare it against using a state-of-the-art Residual Neural Network (ResNet) [7]. Our goal has been to investigate the performance of ViTs when learning to associate depth information to landing zones, whose depth maps are expected to show regular patterns in the depth values as a result of the uniformity of the terrain, which makes it good for a landing zone.

*Email address(es): victoria.vazquez@inaoep.mx, carranza@inaoep.mx

http://www.imavs.org/

We conducted our experiments using the dataset ESPADA [5], which provides pairs of RGB and depth aerial images. We also evaluated our approach using real images captured with a drone. Our results indicate that ViTs outperform ResNet requiring even less training data, making them a robust and effective approach to address the landing zone detection problem. Similarly, we also carried out experiments using depth images captured with the OAK-D camera onboard a MAV while flying indoors (see Fig. 1, confirming that the ViTs, when trained with a small dataset, outperforms ResNet in accuracy and slightly in operation frequency.

In order to present our approach, this paper has been organised as follows: the related work is presented in Section 2; our methodology is described in Section 3; the experimental framework is presented in Section 4 and finally, our conclusion are drawn in Section 5.

2 RELATED WORK

The increasing use of autonomous MAVs has created a need for safe landing zone (SLZ) detection techniques[8, 9], particularly in scenarios where MAVs experience technical difficulties, such as low battery, adverse weather conditions, mechanical failures, and interference. In these situations, the MAVs have an urgent need to land safely. The issues surrounding MAV landing has attracted wide attention, thus resulting in the development of two main detection techniques: non-vision-based [10, 11, 12] and vision-based[13, 14, 6].

Vision-based techniques offer distinct advantages, including strong autonomy, cost-effectiveness, and robust anti-interference capabilities. These techniques encompass various approaches: *(1) Camera-based techniques*, which utilise different configurations of cameras, such as monocular, stereo ranging, or multi-camera setups.[15, 16] *(2) Structure from Motion (SfM)* involves processing a sequence of 2D images to create a 3D terrain surface, enabling the estimation of orientation, photo location, and camera parameters such as focal length, radial, and tangential distortion. *(3) Colour segmentation* is based on grayscale thresholding to identify relevant features. *(4) Simultaneous Localisation and Mapping (SLAM)* estimates MAV localisation by generating a 3D map of the environment and using it to guide the landing [17]. Currently, due to the demonstrated capabilities of deep learning, camera-based techniques can be integrated into computer vision classification algorithms for detecting landing zones in MAV applications.

Deep convolutional neural networks (DCNNs) have led to significant breakthroughs in image classification. [18, 19] DCNNs naturally integrate low, mid, and high-level features and classifiers and can extract different features due to the number of stacked layers [20]. To date, researchers have proposed several CNN architecture models, including VGG, LeNet, GoogleLeNet, and AlexNET. We selected ResNet-50 for CNN image classification due to its advantages in addressing network degradation, and compared to GoogLeNet and VGG, ResNet offers greater computational efficiency and lower time complexity [21].

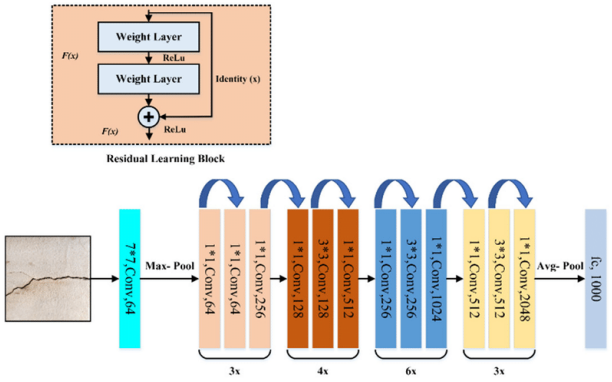


Figure 2: ResNet-50 model architecture. Image taken from [23]

ResNet [7] emerged from inquiring whether stacking more layers would enable the model to learn more effectively. To address the problem of vanishing/exploding gradients, the ResNet model introduced Residual Blocks. Using skip connections, the model connects activations from one layer to subsequent layers, forming residual blocks. These blocks are then stacked together, resulting in a powerful image classification model that can be trained on large datasets and achieve state-of-the-art results.

On the other hand, the Transformer model has emerged as a deep neural network initially applied in natural language processing (NLP) tasks, primarily based on the self-attention mechanism. Due to the Transformer's success in NLP, there has been increasing interest in applying it to computer vision tasks such as classification. This led to developing the Vision Transformer model (ViT) [22]. Recent studies, such as [22, 1], have demonstrated that ViT achieves excellent results compared to state-of-the-art convolutional networks while requiring substantially fewer computational resources for training.

3 METHODOLOGY

Due to the State-Of-The-Art (SOTA) performance of pre-trained ViT models compared to CNN models in image classification tasks, we aim to take advantage of the Vision Transformers' attention-based mechanism over small datasets and noisy depth images for safe landing zone detection. Therefore, in this section, we describe the architecture of the ResNet and ViT models and the dataset we used to fine-tune both models.

3.1 Deep Residual Network (ResNet)

We implemented the ResNet-50 model pre-trained on $ImageNet1K_V2$, see Fig. 2. The ResNet-50 architecture consists of the following main components: **Input Layer:** This layer receives an image of size 224x224x3. It includes

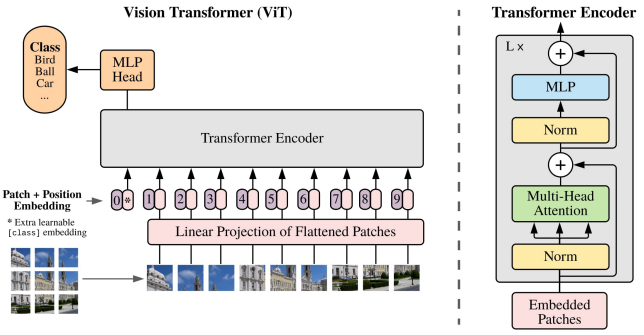


Figure 3: ViT architecture. Taken from [22]



Figure 4: Bebop Parrot 2 with drone equipment specifications.

a 7x7 convolutional layer with a stride of 2, followed by batch normalisation and a ReLU activation function. This layer extracts basic features from the input image and is followed by a 3x3 max-pooling layer, also with a stride of 2, which downsamples the output. **Residual Blocks:** Include bottleneck blocks, each with three convolutional layers (1x1, 3x3, 1x1). The 1x1 convolutions reduce and restore the dimensions, while the 3x3 convolution processes the spatial features. Each convolutional layer is followed by batch normalisation and a ReLU activation function. Shortcut connections add the input of the block directly to the output, allowing the network to learn residual functions and mitigating the vanishing gradient problem. **Fully Connected Layer:** This layer maps the output to the desired number of classes using a softmax activation function.

3.2 *Vision Transformer*

The Vision Transfomer (ViT) model takes the height and width dimensions of the input image as parameters (H and W, respectively) and works with a specified number of channels (C). The ViT transformer divides the input image into N patches with a resolution of (P, P), where the number of patches is defined as:

$$N = HW/P^2 \tag{1}$$

This N value serves as the effective input sequence length for the Transformer. The ViT model maintains a constant latent vector size D, where patches are flattened and mapped to D dimensions. Each image patch is tokenized and processed by a Transformer encoder, which represents the image. The attention head is implemented by a multi-layer perceptron (MLP) with one hidden layer. Position embeddings are added to the patch images to retain positional information. The transformer encoder consists of alternating multi-head attention layers and MLP blocks, see Fig. 3.

3.3 *Extended Synthetic and Photogrammetric Aerial-Image Dataset (ESPADA)*

We used the ESPADA dataset [5] to train ResNet and ViT classification models. ESPADA is an aerial image dataset to train deep neural networks for depth image estimation from a single aerial image.

ESPADA contains image pairs comprising chromatic images and their corresponding depth images, created from synthetic scenes and photogrammetric models imported into the AirSim simulator [24]. The camera view is top-down, and the image resolution for both RGB and Depth images is 640x480 pixels, with a field of view of 80°. Image overlap ranges between 80 and 90 per cent, depending on the drone's height. The dataset includes 80,000 RGB-D image pairs across 49 scenes: 35 from photogrammetric models and 14 from synthetic scenes, divided into five categories: urban, neighbourhood, rural, agriculture, and field. ESPADA is pre-split into training and evaluation datasets. Additionally, it contains three aerial video sequences captured with a drone flying over real scenes, acquired from heights ranging from 30 to 120 metres. For our purposes, we used solely images generated with photogrammetric models as it is argued by the authors that these images provide a more photorealistic and closer resemblance to real aerial images [5].

3.4 *Oak-D dataset*

Based on [25] framework, we used the Parrot Bebop 2 drone equipped with an Oak-D sensor, as the drone's camera is monocular, and an Intel Compute Stick as the host, depicted in Fig. 4. Additionally, the Bebop drone cannot capture top-down views as required due to the absence of a gimbal. To address this limitation, we positioned the Oak-D sensor at the front of the vehicle, facing downwards, to capture top-down views. We reduced the weight of the Oak-D sensor by replacing the camera case with a 3D-printed piece to avoid instabilities due to the Oak-D camera position, as it was done as well in [25].

The depth camera has a minimum range of 0.40 metres and a maximum range of approximately 8 metres. The Oak-D dataset contains 750 paired depth and monocular images at a resolution of 640x480 pixels collected from four flight sequences captured at heights between 0.50 metres and 2 metres. These sequences are divided into four categories: Desk,

http://www.imavs.org/

http://www.imavs.org/



Figure 5: Examples of aerial images and their corresponding estimated depth used for training. Note that planar surfaces exhibit a similar red tone, ideal for landing. In the depth images, the farther the depth for the camera, the lighter the red colour.

Box, Office, and Hall. The first two datasets consist of sequences flying over obstacles corresponding to their names, with 131 and 55 images, respectively. The last two datasets comprise sequences recorded in an Office (369 images) and a Hall (195 images).

4 EXPERIMENTAL FRAMEWORK

In this section, we describe the evaluation results using two different datasets: ESPADA and our small depth image dataset captured with an Oak-D sensor mounted on the drone as described in Section 3.

For each dataset, we conducted two evaluations: one with fewer images and another with a larger number. No data augmentation techniques were applied. This approach was taken to assess the impact of training the neural models with limited data, and with noisy data in the case of the images captured with the OAK-D sensor. This is particularly relevant in scenarios where there is insufficient time to collect extensive data, yet it is essential to train a model that can effectively perform tasks such as landing zone detection, even with the limited data available.

For the two experiments conducted with the ESPADA dataset, we selected sets of 200 and 400 depth images evenly split between landing and no-landing zones. Examples of these images are shown in Fig. 5. The model was trained using an 80-20% split for training and testing, respectively; we show two accuracy columns in all the tables: 'Training accuracy' was calculated immediately using a 20% split of the images for evaluating, while the other accuracy column, 'Test accuracy', was obtained by comparing the model's predictions against 150 unseen labelled images. It is important to note that these 150 unseen images were selected from the ES-

http://www.imavs.org/

PADA dataset, specifically from the validation image subsets from the five categories in ESPADA, all captured at a height of 30 metres.

Furthermore, we conducted two additional experiments. This time, the ResNet and ViT models were trained with two small datasets taken from the Oak-D dataset, with 84 and 284 images. Examples of these images are shown in Fig. 6. As before, we selected 80% for training and 20% for testing. We evaluated the model against 120 unseen labelled images for 'Test accuracy'. It's noteworthy that the images from the Oak-D camera were noisier compared to those from the ESPADA dataset.

Experiments conducted with the ESPADA dataset, summarised in Table. 1 and 2, reveal the following observations. For the ResNet model, due to the limited information in the 200-image dataset, the model achieves its highest accuracy within the first 20 epochs. However, as the number of epochs increases, accuracy decreases, likely because the model falls into a local minimum. This behaviour was consistently observed in most of the training runs for the ResNet model. On the other hand, the ViT model, with the same 200-image dataset, reaches higher accuracy in 20 epochs and still increases its accuracy.

Experiments conducted with the 400-image dataset showed better performance for the ResNet model compared to the when using a smaller dataset, as ResNet struggles with limited information. However, the ViT model still achieved superior results when evaluated using real images.

To evaluate the performance of both models in environments different from those represented in ESPADA, we captured a dataset in an indoor scenario using the OAK-D camera onboard a MAV, as described in Section 3. In Table 3, the first row shows the results of training the ResNet model with the Oak-D small dataset of 84 images and its corresponding accuracy. We conducted this training multiple times and selected the model with the best training accuracy at 50 epochs. This selection was made because increasing the number of epochs led to a decrease in accuracy or caused the accuracy to oscillate. For the second row, with more images for training, we can observe the ResNet model performance is better.

Similar to the ResNet experiment, in Table 4, the first row shows the results of training the ViT model with the Oak-D small dataset of 84 images and its corresponding accuracy. Both training and test accuracies are reported at 50 epochs. The ViT model demonstrates faster convergence during training and performs better than the ResNet model trained with the same small number of examples. The ViT model can reach 80% accuracy within just 15 to 20 epochs.

Finally, in terms of processing time, the ResNet model runs at 8.3 ms, while the ViT model runs at 11.9 ms on average, both on outdated GPU hardware (GeForce GTX 1070) using CUDA 12.2. This is encouraging for future work, as we aim to run these models on low-budget processors. We note that, due to library issues and time constraints, we were unable to import the models to run on the OAK-D camera, which has a GPU. However, we plan to pursue this effort to fully utilise the sensor's capabilities.

ResNet Trained on ESPADA			
Images	**Epochs**	**Training Acc**	**Test Acc(%)**
200	20	51.28	**57.27**
	30	53.85	32.7
	40	62.12	30.5
	50	70.9	30
400	20	68.18	67.27
	30	70.9	70.0
	40	84.62	84.54
	50	87.18	**86.9**

Table 1: Accuracy of the fine-tuned ResNet-50 model trained with ESPADA. Note that the model requires more images and more epochs to achieve more accuracy, highlighted in bold. A total of 100 images were used as test set.

ViT Trained on ESPADA			
Images	**Epochs**	**Training Acc**	**Test Acc(%)**
200	20	86.67	81.9
	30	85.85	83.63
	40	89.12	86.09
	50	91.9	**87.0**
400	20	80.9	79.8
	30	86.36.0	82.36
	40	96 .36	90.9
	50	96.45	**91.3**

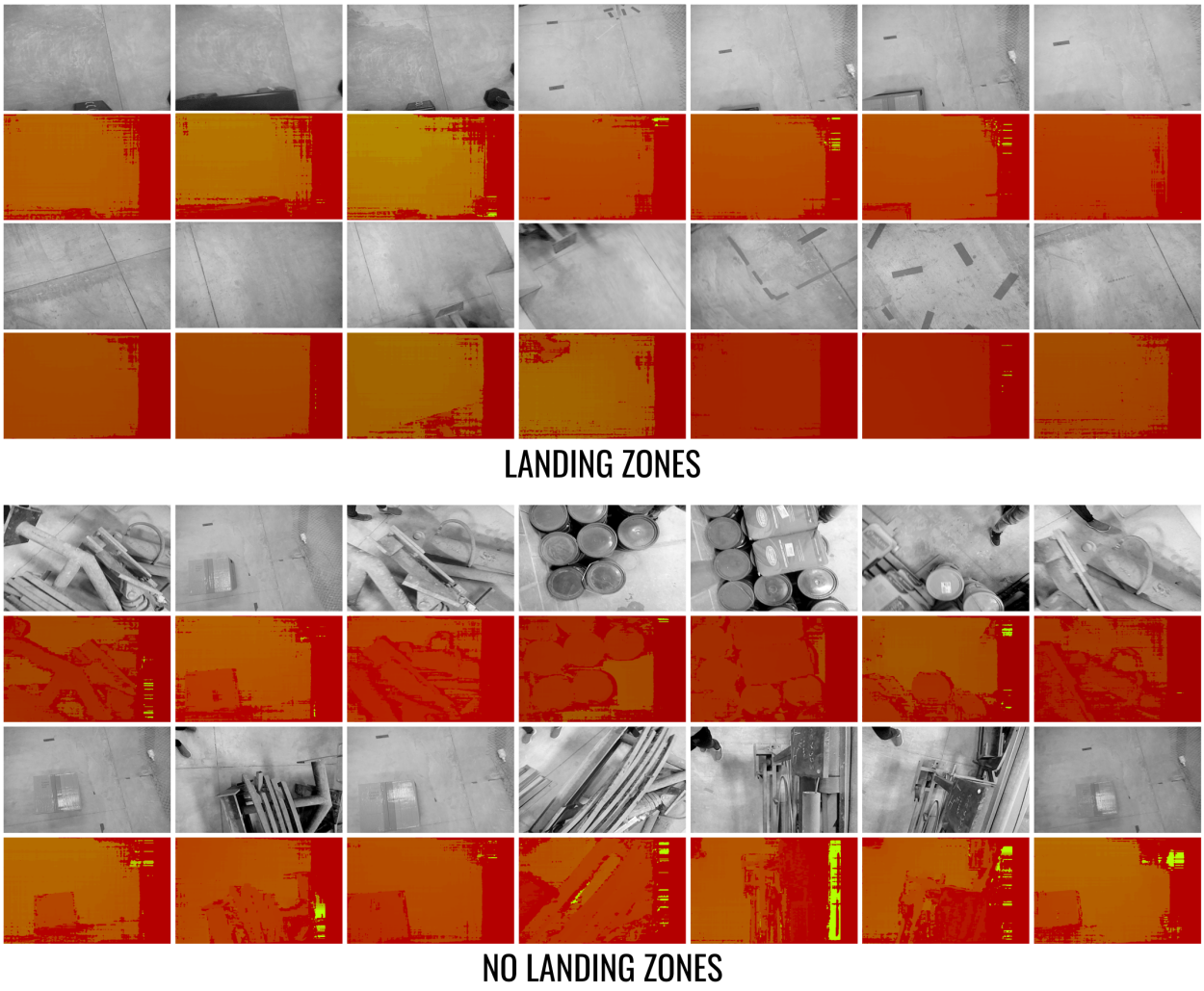
Table 2: Accuracy of the ViT model trained with ESPADA. Note that the model achieves superior results with fewer images compared to ResNet, and the best results when using more images for training, highlighted in bold.

ResNet Trained on captured dataset			
Images	**Epochs**	**Training Acc**	**Test Acc(%)**
84	50	70.9	62.4
284	50	94.9	**96.6**

Table 3: Accuracy for the ResNet model trained with images captured with the OAK-D camera onboard the MAV.

5 CONCLUSION

We have presented a methodology to address the problem of landing zone detection for MAVs. To achieve this, we have explored the use of two popular Deep Neural Networks,



http://www.imavs.org/

Figure 6: Examples of aerial images taken from our dataset and their corresponding estimated depth used for training. Note that planar surfaces, ideal for landing, exhibit a similar red tone, and Oak-D images are noisier than ESPADA dataset. In the depth images, the farther the depth for the camera, the lighter the red colour. Taken from [5]

ViT Trained on captured dataset			
Images	**Epochs**	**Training Acc**	**Test Acc(%)**
84	50	82.7	73.63
284	50	98.2	**96.36**

Table 4: Accuracy of the ViT model trained with images captured by the OAK-D camera onboard the MAV. Note that this model achieved better results than ResNet with fewer images.

namely Residual Networks and Vision Transformers. The former is a Convolutional Neural Network that has proven highly successful in various computer vision tasks. However, the latter offers an architecture capable of capturing spatial relations among visual data, thanks to its attention mechanism. Therefore, for computer vision tasks, it may offer superior performance when trained with fewer examples.

These findings were confirmed in our experiments, where we trained both models using aerial images from a state-of-the-art dataset known as ESPADA and images captured with the OAK-D sensor, a depth camera mounted on a MAV. We opted to use depth images in both cases, as the literature suggests that depth images provide richer data in the form of depth values, aiding in determining whether the observed scene corresponds to a landing zone [6].

Although CNNs have a strong track record in various computer vision tasks and are efficient with large-scale

datasets, we have also explored the use of both models trained from scratch and pre-trained models. The choice to employ Vision Transformers (ViTs) and to conduct a more in-depth analysis is grounded in previous studies that contrast ViTs and CNNs [26, 27, 28]. Based on these works and our experiments, our results indicate that pre-trained Vision Transformers excel in scenarios where understanding global dependencies and context is crucial. Vision Transformers typically require larger amounts of training data to achieve comparable performance to CNNs. Our findings reinforce that the ViT model surpasses the ResNet model, particularly when pre-trained, resulting in better accuracy when fine-tuned with fewer images. It is also noteworthy that the depth images from the OAK-D camera were less accurate than those from other depth cameras, such as Intel's RealSense. Yet, the ViT model effectively managed the noisy data. Moreover, with an average prediction time of $11.9ms$ on an outdated GPU, the ViT model shows potential for real-time detection when implemented on state-of-the-art processors aboard the vehicle.

Future work will involve testing depth images generated by a Deep Neural Network from a single image, as outlined in [29]. Additionally, we will explore distillation strategies for the ViT to enhance prediction speed on embedded hardware. Furthermore, we will investigate Explainable Artificial Intelligence methods to provide a more in-depth analysis of feature extraction characteristics in both ViT and CNN models.

http://www.imavs.org/

ACKNOWLEDGEMENTS

The first author is thankful to the National Council of Science and Technology of Mexico (CONAHCYT) for her scholarship number 1026123.

REFERENCES

[1] Y. et al. Liu. A survey of visual transformers. *IEEE Transactions on Neural Networks and Learning Systems*, 35(6):7478–7498, June 2024.

[2] Leticia Oyuki Rojas-Perez and Jose Martinez-Carranza. Towards autonomous drone racing without gpu using an oak-d smart camera. *Sensors*, 21(22):7436, 2021.

[3] L Oyuki Rojas-Perez and Jose Martinez-Carranza. Deeppilot4pose: a fast pose localisation for mav indoor flight using the oak-d camera. *Journal of Real-Time Image Processing*, 20(1):8, 2023.

[4] M. Hermann, B. Ruf, M. Weinmann, and S. Hinz. Self-supervised learning for monocular depth estimation from aerial imagery. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-2-2020:357–364, 2020.

[5] Rafael Lopez-Campos and Jose Martinez-Carranza. Espada: Extended synthetic and photogrammetric aerial-image dataset. *IEEE Robotics and Automation Letters*, 6(4):7981–7988, 2021.

[6] L Oyuki Rojas-Perez, Roberto Munguia-Silva, and Jose Martinez-Carranza. Real-time landing zone detection for uavs using single aerial images. In *10th international micro air vehicle competition and conference, Melbourne, Australia*, pages 243–248, 2018.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] J. Martinez-Carranza, E. Inzunza-Gonzalez, E. E. Garcia-Guerrero, and E. Tlelo-Cuautle. *Machine Learning for Complex and Unmanned Systems*. CRC Press, 2024.

[9] M. S. Alam and J. Oluoch. A survey of safe landing zone detection techniques for autonomous unmanned aerial vehicles (uavs). *Expert Systems with Applications*, 179:115091, 2021.

[10] J. Lim, M. Kim, H. Yoo, and J. Lee. Autonomous multirotor uav search and landing on safe spots based on combined semantic and depth information from an onboard camera and lidar. *IEEE/ASME Transactions on Mechatronics*, 2024.

[11] F. Neves, L. Branco, M. Pereira, R. Claro, and A. Pinto. A multimodal learning-based approach for autonomous landing of uav. *arXiv preprint arXiv:2405.12681*, 2024.

[12] J. Chen, W. Du, J. Lin, U. M. Borhan, Y. Lin, B. Du, and J. Li. Emergency uav landing on unknown field using depth-enhanced graph structure. *IEEE Transactions on Automation Science and Engineering*, 2024.

[13] 2014 International Conference et al. Vision-based autonomous landing system for unmanned aerial vehicle: A survey. In *2014 International Conference on Multisensor Fusion and Information Integration for Intelligent Systems (MFI)*, pages 1–8. IEEE, 2014.

[14] Aldrich A Cabrera-Ponce and Jose Martinez-Carranza. A vision-based approach for autonomous landing. In *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, pages 126–131. IEEE, 2017.

[15] C. Theodore, D. Rowley, A. Ansar, L. Matthies, S. Goldberg, and D. Hubbard. Flight trials of a rotorcraft unmanned aerial vehicle landing autonomously at unprepared sites. In *Annual Forum Proceedings-American Helicopter Society*, page 1250. AMERICAN HELICOPTER SOCIETY, INC, 2006.

[16] M. Meingast, C. Geyer, and S. Sastry. Vision based terrain recovery for landing unmanned aerial vehicles. In *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No. 04CH37601)*, pages 1670–1675. IEEE, 2004.

[17] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics & Automation Magazine*, 13(2):99–110, 2006.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[19] Yann LeCun, Bernhard Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1989.

[20] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional neural networks. In *European Conference on Computer Vision (ECCV)*, 2014.

[21] YuYu Zheng, HaoXuan Huang, and Junming Chen. Comparative analysis of various models for image classification on cifar-100 dataset. *Journal of Physics: Conference Series*, 2711:012015, 2024.

[22] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, ..., and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[23] Luqman Ali, Fady Alnajjar, Hamad Jassmi, Munkhjargal Gochoo, Wasif Khan, and Mohamed Serhani. Performance evaluation of deep cnn-based crack detection and localization techniques for concrete structures. *Sensors*, 21:1688, 2021.

[24] S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics: Results of the 11th International Conference*, pages 621–635. Springer International Publishing, 2018.

[25] L. O. Rojas-Perez and J. Martinez-Carranza. Towards autonomous drone racing without gpu using an oak-d smart camera. *Sensors*, 21(22):7436, 2021.

[26] C. Matsoukas, J. F. Haslum, M. Söderberg, and K. Smith. Is it time to replace cnns with transformers for medical images? *arXiv preprint arXiv:2108.09038*, 2021.

[27] Y. Bai, J. Mei, A. L. Yuille, and C. Xie. Are transformers more robust than cnns? *Advances in Neural Information Processing Systems*, 34:26831–26843, 2021.

[28] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM Computing Surveys*, 54(10s):200, January 2022.

[29] Luis Pellegrin and José Martinez-Carranza. Towards depth estimation in a single aerial image. *International journal of remote sensing*, 41(5):1970–1985, 2020.

http://www.imavs.org/

Crossing Gate Detection Using Audio Frequency Pitch Tracking

Caleb Rascon[1*], L. Oyuki Rojas-Perez[2], Victoria Eugenia Vazquez-Meza[2], Aldrich A. Cabrera-Ponce[3] and Jose Martinez-Carranza[2*]
[1]Instituto de Investigaciones en Matematicas Avanzadas y en Sistemas, Universidad Nacional Autonoma de Mexico.
[2]Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), Puebla, Mexico.
[3]Benemérita Universidad Autónoma de Puebla (BUAP), Puebla, Mexico.

http://www.imavs.org/

ABSTRACT

Autonomous Drone Racing (ADR) has garnered significant interest in the aerial robotics community. Early solutions used classical computer vision algorithms for gate detection, while more recent approaches employed visual Simultaneous Localisation and Mapping (SLAM). The latest advancements have showcased solutions capable of winning races against world champions. However, these rely primarily on visual data from onboard cameras, whereas humans complement visual sensing with auditory perception. Motivated by the benefits of auditory perception, this study investigates the use of audio signal processing to detect when a drone crosses a gate during a race. This detection addresses the blind spot issue, where the gate disappears from the visual sensor's view after crossing. Initial results indicate the feasibility of using audio signals to identify gate crossings, based on sound changes caused by drone propellers. This is a first effort to explore the broader potential of auditory perception in autonomous drone racing.

1 INTRODUCTION

Autonomous Drone Racing (ADR) is a scientific challenge that has garnered significant interest within the aerial robotics community. Initially, solutions relied on classical computer vision algorithms for gate detection to guide the drone's flight towards the gate [1]. Subsequent competitions incorporated more sophisticated techniques such as visual Simultaneous Localisation and Mapping (SLAM) [2]. Recently, impressive results were achieved in [3], demonstrating a comprehensive solution that won several races against two world champions.

Despite these advancements, the best solutions to date rely primarily on visual data captured by onboard cameras. While impressive, humans do not rely solely on visual sensing to navigate through the flying arena. Our sense of surroundings is complemented by auditory perception, which, in the context of a drone race with several adversaries, is used to infer when other drones fly by. Furthermore, trained pilots use auditory perception to detect malfunctioning motors or propellers.

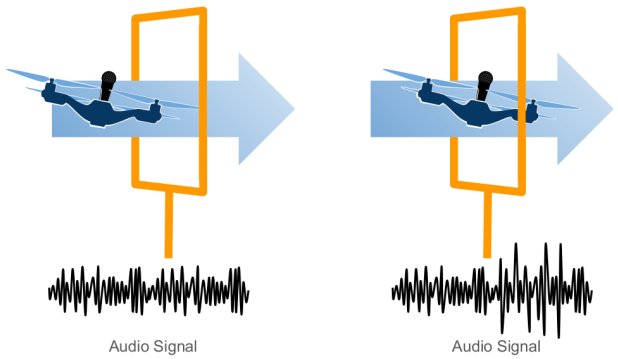


Figure 1

Motivated by the advantages of auditory perception, we present a preliminary study on how audio signal processing could be used for a specific task in autonomous drone racing: detecting when the drone crosses the gate during the race. Detecting gate crossings can be highly beneficial for designing navigation policies for autonomous drone racers. If a policy relies on vision for gate detection, the gate will eventually move out of the visual sensor's view once the drone crosses it. However, this can also occur if the drone navigates off the planned trajectory, flying past the gate but not through it.

The moment the drone flies towards the gate and crosses it—when the gate disappears from the visual sensor's field of view—is what we call the blind spot. Given the importance of detecting this blind spot, we explore the use of audio signals to detect when the drone crosses the gate. This is based on the observation that there is a slight change in sound due to the wind produced by the drone's propellers hitting the gate structure.

Our initial results indicate that this detection is possible. We acknowledge that this detection might be specific to our experimental setup, including the type of drone and gate used. Nevertheless, it is intriguing to see that audio signals can be utilised in this context. Further research could explore the potential of auditory perception in autonomous drone racing

*Email address(es): carranza@inaoep.mx, caleb@unam.mx

http://www.imavs.org/

2 RELATED WORK

Drone racing is a challenging benchmark to gauge progress in complex perception, planning, and control algorithms, as drones must be able to perceive and interpret the scene, plan, and act within milliseconds [2, 4, 5]. In a racing setting, the quadrotor must fly through a sequence of gates in the minimum time without collisions. Two strategies are proposed to solve this challenge.

The first strategy uses external sensors, such as a motion capture system, and assumes a controlled environment with privileged knowledge. This means that the control system has full access to the drone's state (position and orientation) and the number and state of the gates, which makes it possible to implement a time-optimal planner and a flight controller based on MPC [6, 7]. Nevertheless, this is a partial solution to drone racing because the perception task is removed to complete the racetrack at high speed.

The second strategy uses only onboard sensors for state estimation and gate identification [2, 8, 9, 10], the latter employing techniques based on Deep Learning used for object detection, such as a Single Shot Detector (SSD) [11, 12] or YOLO [13]. However, systems based on visual information face a challenge when the drone crosses the gate because the gate disappears from the field of view, making it impossible for the system to determine if the drone is in the middle of the gate or outside. This issue is referred to as a blind spot. Some solutions have utilised different sensors to overcome this problem, such as a camera to see the gate's base [8, 9], a LIDAR to measure changes between the drone and the ground [1], an IMU to estimate the drone's displacement [14, 15], or visual temporal information to identify the entrance and exit of the gate [16, 12, 17].

This paper proposes a novel approach based on acoustics to assist in blind spot detection. By incorporating audio sensors, the system can detect subtle changes in sound that indicate the drone's position relative to the gate, providing an additional layer of information that visual sensors might miss. Auditory perception offers several opportunities. Audio sensors have introduced auxiliary features that can be useful for drone navigation. In [18], the authors provide an extensive analysis of works that have reported using microphones to capture audio signals and the applications where drones are involved. For example, avoid collisions [19], sound source localisation [20, 21], detection of propeller anomalies [22], intruder detection [23, 24], classification [25, 26, 27], and drone proximity [28]. This motivates the analysis of noise production and mitigation [29].

3 PROPOSED METHODOLOGY

When a drone is passing through a gate, its height sensor will briefly sense the gate, which may cause the drone to assume that the floor has suddenly "jumped" up which, in turn, may cause its flight control framework (that has the task of maintaining a given altitude) to incorrectly fly upward. To avoid this, it is common for drones' flight control frameworks to ignore high-variance changes in the height sensor values. Additionally, since the height sensor sampling frequency and/or the frequency of the feedback loop are usually relatively low (¡ 5 Hz.), this issue may not even be contemplated.

In any case, it is frequently observed in flight control frameworks that the drone's altitude is briefly "corrected" to counter this sudden change in height. During this momentary correction, the drone's motors can be heard being briefly engaged. In Figure 2, it can be seen how the pitch (reported in Hertz in the vertical axis) changes through time (horizontal axis). It can also be seen how the pitch of the drone's motors presents sudden but major changes while crossing the gate (this moment is presented as a vertical black line[1]).

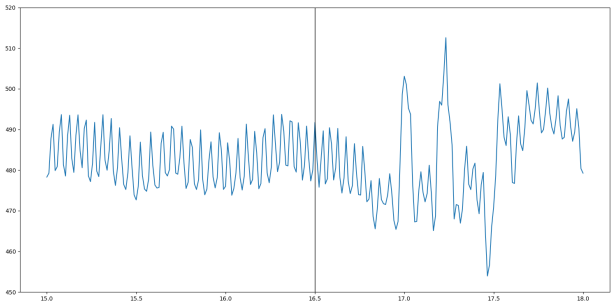


Figure 2: Drone's motor pitch through time. Black vertical line: a rough estimation of when the drone crossed the gate.

In this work, the pitch of an audio signal refers to its highest-magnitude frequency. The pitch reported in Figure 2 was estimated by first carrying out the Short-Time Fourier Transform (STFT, with a FFT window length of N samples and hop length of H samples) of L_w seconds of the audio recording of the drone's noise. Since the pitch is the main focus of the proposed technique, it is important that its estimation is not limited by the frequency binning of the discrete Fast Fourier Transform (FFT). Thus, the maximum value of each FFT window of the resulting STFT'ed signal was refined using the popular quadratically-interpolated FFT technique (QIFFT) [30]. This process finds a closer value of the true peak, between frequency bins, by extrapolating a parabola using the frequency bin of the maximum value of the FFT window and its two neighbouring frequency bins. The result is similar to using a maximum-likelihood estimator, since a parabola is a Gaussian window in the decibel scale.

The proposed technique is as follows:

Every L_h seconds, the pitch information ($[p_1, p_2, \ldots, p_K]$) is estimated from each of the K FFT windows of the current L_w seconds of captured audio. Then, it is established if the estimated pitch of the last FFT

[1] This is a rough estimation, based on manually checking the video of the drone's flight.

Algorithm 1 Pseudo-code for proposed technique.

K := number of FFT windows from STFT
$\bar{p}$:= mean pitch of the last L_w window
σ_p := pitch variance of the last L_w window
for every L_h seconds **do**
  X := STFT of current L_w seconds
  X_k := kth FFT window of X
  for $k = 1$ **to** K **do**
    $p_k \leftarrow max(QIFFT(|X_k|))$
  end for
  if $|p_K - \bar{p}| > M_\sigma \sigma_p$ **then**
    DRONE IS CROSSING GATE
  end if
  $\bar{p} \leftarrow \left[\sum_{k=1}^{K} p_k\right] / K$
  $\sigma_p \leftarrow \sqrt{\left[\sum_{k=1}^{K} (p_k - \bar{p})\right] / K}$
end for

window (p_K) has suddenly considerably changed from past pitches. This is the case if the difference between p_K and the mean pitch of the last L_w seconds ($\bar{p}$) is greater than its pitch variance (σ_p), multiplied by a sensitivity factor (M_σ). Finally, $\bar{p}$ and σ_p are re-calculated with the pitch information of the current L_w seconds to be used for the next capture window.

An important virtue of the proposed technique is that its response time can be calibrated by establishing the value of L_h. The lower limit of this value is the processing time of Algorithm 1. This time, in turn, depends on the processing time of carrying out the STFT and QIFFT, both of which are quite efficient. However, to allow for real-time operation, the reduction of the value of L_h should be accompanied with a reduction of L_w, since the STFT operation depends on it, as well as the $k \in [1, K]$ loop in Algorithm 1. Reducing the value of L_w provides less past pitch information, making the pitch change less predictable. The values of L_h and L_w presented in the next section provide a good balance between response time and predictable behaviour.

It may be tempting to change Algorithm 1 so that, instead of calculating the STFT of the whole L_w audio window every L_h seconds, we only calculate the FFT of the last FFT window. Doing so would reduce the processing time considerably. However, this would be equivalent on carrying out the STFT with no overlap between FFT windows ($H = 0$ samples), which would result in a highly varying pitch-over-time signal. Carrying out the STFT of the whole L_w audio window results in a much smoother pitch transition through time, which simplifies detecting sudden pitch changes. Moreover, the LibROSA [31] implementation of both the STFT and QIFFT was used (via the *piptrack* function), which provided very low response times (~ 0.002 seconds to process $L_w = 1.5$ seconds of audio in hardware of moderate computing power). Having established all of this, it would be of interest (and left for future work) to further lower the response time of Algorithm 1 by only updating $\bar{p}$ and σ_p using only the current FFT window to model a smooth pitch-over-time signal.



Figure 3: Drone and hardware employed in our experimental framework. We used a wireless microphone transmitting to the GCS, an Alienware 15 R3 laptop.

It is important to mention that this technique is aimed to be used once a gate crossing is expected in the near future of the planned flight; no further flight adjustments should be carried out once this technique is triggered. This is because a change in pitch can occur mid-flight for reasons other than a gate crossing. As a reminder, the advantage of this technique is its potentially low response time compared to using other light-based sensors, which (as previously described) is highly configurable given different scenarios.

4 EVALUATION AND RESULTS

For our experiments, we used the Bebop 2.0 Power Edition drone from Parrot. We attached a Bluetooth microphone to this drone, which transmitted to the Ground Control Station (GCS) - an Alienware 15 R3 laptop equipped with a GeForce GTX 1070 Graphics Processing Unit. The microphone weighs 80g and can transmit over a distance of up to 20 metres, see Fig. 3.

Several recordings were captured for testing, where the drone was given a linear route that passes through a gate. A microphone was connected to the onboard computer to record the audio. Additionally, a video recording was also captured during the flight to provide an estimate of when the drone passed through the gate.

In our tests, we used the following configuration that allows for real-time operation, with a relatively low response time:

In Figure 5, three runs are shown using the proposed technique to detect the crossing of the gate. Similar to Figure 2, a vertical black line shows a rough estimation of the moment when the drone is crossing the gate. To facilitate the visualisation of the results, an orange line reports the gate crossing

http://www.imavs.org/

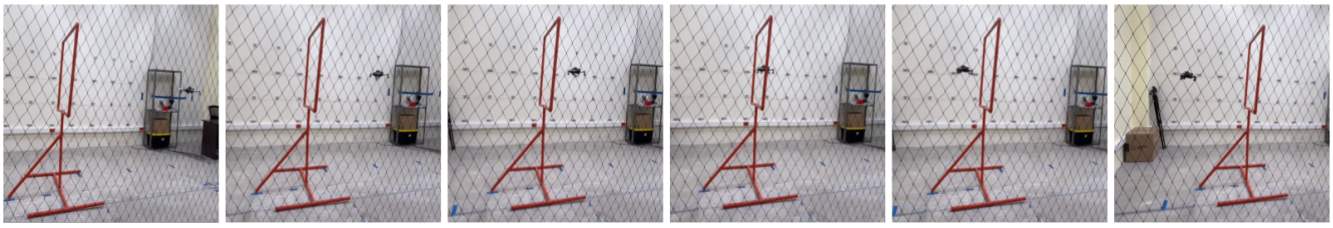


Figure 4: From left to right: image shots illustrating one of the runs in the lab where the drone flew through a square gate with a width of 1m and 2m in height. The drone has a microphone attached on board transmitting the audio signal to the GCS.

L_w	1.5 seconds
L_h	0.1 seconds
M_σ	2
N	2048 samples
H	512 samples
f_s	48000 samples per second
K	$L_w f_s / H \rightarrow 140$ FFT windows

Table 1: Configuration values used in tests.

detection results in the form of a peak.

As it can be seen, a peak is present when the pitch has changed dramatically from the value of its predecessors, clearly marking the moment when the drone's motors were briefly adjusted. This, in turn, can be assumed as being the moment in which the drone is crossing the gate.

It can also be seen how the technique picks up other pitch changes after the gate crossing, probably due to the drone's motor changing speed during the drone's landing procedure. This recalls what was mentioned earlier of how this technique should only be used when the gate crossing is eminent in the drone's flight plan.

5 CONCLUSION

In this work, we have presented initial results on the use of audio signal processing to detect when a drone flies through a gate in the context of autonomous drone racing. We used an off-the-shelf Bluetooth microphone that transmits the audio signal to the Ground Control Station (GCS). As shown in our experiments, our methodology enables the detection of spikes in the audio signal triggered by the drone's propellers when it crosses the gate. This sound is similar to that produced by the propellers during take-off or when switching to forward motion after hovering, and likewise when stopping and returning to hovering. However, it is notable that this similar sound is produced when crossing the gate.

We argue that when crossing the gate, the drone detects the change in height from the floor to the horizontal bar of the gate with its altimeter sensor, and its internal controller attempts to maintain horizontal flight to keep moving forward. This slight compensation produces the change in sound that can be detected with our methodology.

Although detecting the gate crossing could be done with the altimeter alone, audio processing could act as a complementary measurement, hence our interest in investigating this approach.

Aware of the specificity of our experimental scenario, in our future work, we plan to conduct additional experiments with various gate types and different drones. Furthermore, we will assess whether the changes in the audio signal are caused by the gate's aerodynamics or result from the Bebop's internal control system, where the altitude sensor detects the base of the gate and adjusts the drone's signals to maintain the required altitude.

ACKNOWLEDGEMENTS

The second, third, and fourth authors are thankful to the National Council of Humanities, Science and Technology of Mexico (CONAHCYT) for their scholarship number 924254, 1026123, and 802791, respectively.

REFERENCES

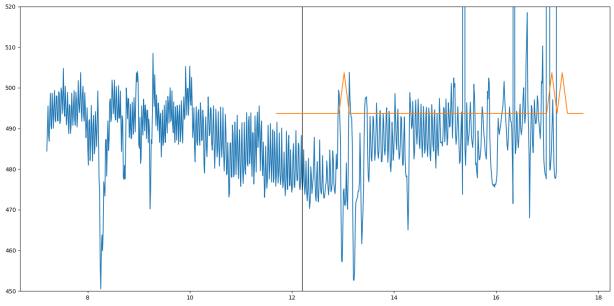
[1] Sunggoo Jung, Sungwook Cho, Dasol Lee, Hanseob Lee, and David Hyunchul Shim. A direct visual servoing-based framework for the 2016 iros autonomous drone racing challenge. *Journal of Field Robotics*, 35(1):146–166, 2018.

[2] Hyungpil Moon, Jose Martinez-Carranza, Titus Cieslewski, Matthias Faessler, Davide Falanga, Alessandro Simovic, Davide Scaramuzza, Shuo Li, Michael Ozo, Christophe De Wagter, et al. Challenges and implemented technologies used in autonomous drone racing. *Intelligent Service Robotics*, 12(2):137–148, 2019.

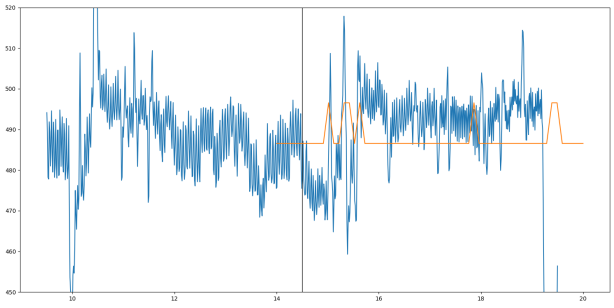
[3] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023.

[4] L Oyuki Rojas-Perez and José Martínez-Carranza. On-board processing for autonomous drone racing: An overview. *Integration*, 80:46–59, 2021.

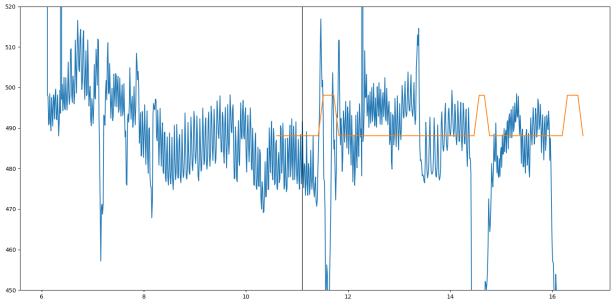
http://www.imavs.org/



(a) Run 1.



(b) Run 2.



(c) Run 3.

Figure 5: Detection results (orange line) plotted over the pitch-over-time signal.

[5] Drew Hanover, Antonio Loquercio, Leonard Bauersfeld, Angel Romero, Robert Penicka, Yunlong Song, Giovanni Cioffi, Elia Kaufmann, and Davide Scaramuzza. Autonomous drone racing: A survey. *IEEE Transactions on Robotics*, 2024.

[6] Philipp Foehn, Dario Brescianini, Elia Kaufmann, Titus Cieslewski, Mathias Gehrig, Manasi Muglikar, and Davide Scaramuzza. Alphapilot: Autonomous drone racing. *Autonomous Robots*, 46(1):307–320, 2022.

[7] Angel Romero, Robert Penicka, and Davide Scaramuzza. Time-optimal online replanning for agile quadrotor flight. *IEEE Robotics and Automation Letters*, 7(3):7730–7737, 2022.

[8] Sunggoo Jung, Sunyou Hwang, Heemin Shin, David Hyunchul Shim, et al. Perception, guidance, and navigation for indoor autonomous drone racing using deep learning. *IEEE Robotics and Automation Letters*, 3(3):2539–2544, 2018.

[9] Sunggoo Jung, Hanseob Lee, Sunyou Hwang, and David Hyunchul Shim. Real time embedded system framework for autonomous drone racing using deep learning techniques. In *2018 AIAA Information Systems-AIAA Infotech@ Aerospace*, page 2138. 2018.

[10] Nitin J Sanket, Chahat Singh, Kanishka Ganguly, Cornelia Fermuller, and Yiannis Aloimonos. Gapflyt: Active vision based minimalist structure-less gap detection for quadrotor flight. *IEEE Robotics and Automation Letters*, 2018.

[11] Aldrich A Cabrera-Ponce, L Oyuki Rojas-Perez, Jesus Ariel Carrasco-Ochoa, Jose Francisco Martinez-Trinidad, and Jose Martinez-Carranza. Gate detection for micro aerial vehicles using a single shot detector. *IEEE Latin America Transactions*, 17(12):2045–2052, 2019.

[12] L Oyuki Rojas-Perez and Jose Martinez-Carranza. Where are the gates: discovering effective waypoints for autonomous drone racing. In *Ibero-American Conference on Artificial Intelligence*, pages 353–365. Springer, 2022.

[13] L Oyuki Rojas-Perez and Jose Martinez-Carranza. Towards autonomous drone racing without gpu using an oak-d smart camera. *Sensors*, 21(22):7436, 2021.

[14] J Arturo Cocoma-Ortega, L Oyuki Rojas-Perez, Aldrich A Cabrera-Ponce, and J Martinez-Carranza. Overcoming the blind spot in cnn-based gate detection for autonomous drone racing. In *2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS)*, pages 253–259. IEEE, 2019.

[15] Shuo Li, Michal MOI Ozo, Christophe De Wagter, Guido CHE de Croon, et al. Autonomous drone race: A computationally efficient vision-based navigation and control strategy. *Robotics and Autonomous Systems*, page 103621, 2020.

[16] L Oyuki Rojas-Perez and Jose Martinez-Carranza. Deeppilot: A cnn for autonomous drone racing. *Sensors*, 20(16):4524, 2020.

[17] L Oyuki Rojas-Perez and Jose Martinez-Carranza. Effective training to improve deeppilot. *AI Communications*, (Preprint):1–18, 2023.

http://www.imavs.org/

[18] Jose Martinez-Carranza and Caleb Rascon. A review on auditory perception for unmanned aerial vehicles. *Sensors*, 20(24):7276, 2020.

[19] Brendan Harvey and Siu O'Young. Acoustic detection of a fixed-wing uav. *Drones*, 2(1):4, 2018.

[20] Koutarou Furukawa, Keita Okutani, Kohei Nagira, Takuma Otsuka, Katsutoshi Itoyama, Kazuhiro Nakadai, and Hiroshi G Okuno. Noise correlation matrix estimation for improving sound source localization by multirotor uav. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3943–3948. IEEE, 2013.

[21] Takuma Ohata, Keisuke Nakamura, Takeshi Mizumoto, Tezuka Taiki, and Kazuhiro Nakadai. Improvement in outdoor sound source detection using a quadrotor-embedded microphone array. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1902–1907. IEEE, 2014.

[22] Sai Srinadhu Katta, Kide Vuojärvi, Sivaprasad Nandyala, Ulla-Maria Kovalainen, and Lauren Baddeley. Real-world on-board uav audio data set for propeller anomalies. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 146–150. IEEE, 2022.

[23] Oscar Ruiz-Espitia, Jose Martinez-Carranza, and Caleb Rascon. Aira-uas: an evaluation corpus for audio processing in unmanned aerial system. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 836–845. IEEE, 2018.

[24] Aldrich A Cabrera-Ponce, J Martinez-Carranza, and Caleb Rascon. Detection of nearby uavs using a multi-microphone array on board a uav. *International Journal of Micro Air Vehicles*, 12:1756829320925748, 2020.

[25] Matthew Ritchie, Francesco Fioranelli, Hervé Borrion, and Hugh Griffiths. Multistatic micro-doppler radar feature extraction for classification of unloaded/loaded micro-drones. *IET Radar, Sonar & Navigation*, 11(1):116–124, 2017.

[26] JJM De Wit, RIA Harmanny, and P Molchanov. Radar micro-doppler feature extraction using the singular value decomposition. In *2014 International Radar Conference*, pages 1–6. IEEE, 2014.

[27] Sara Al-Emadi, Abdulla Al-Ali, Amr Mohammad, and Abdulaziz Al-Ali. Audio based drone detection and identification using deep learning. In *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 459–464. IEEE, 2019.

[28] Qian Zhao, Jason Hughes, and D Lyons. Drone proximity detection via air disturbance analysis. In *Unmanned Systems Technology XXII*, volume 11425, pages 141–149. SPIE, 2020.

[29] Caleb Rascon and Jose Martinez-Carranza. A review of noise production and mitigation in uavs. *Machine Learning for Complex and Unmanned Systems*, pages 220–235, 2024.

[30] Julius O Smith. Parshl: A program for the analysis/synthesis of inharmonic sounds based on a sinusoidal representation. *Stanford Music Department Technical Report, STAN-M-43*, 1987.

[31] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8, 2015.

http://www.imavs.org/

Comparative Analysis of Cameras for ArUco Marker Recognition in Unmanned Aerial Vehicles

Souza, J. V. N. [*], Sobrinho Junior, D. F. [†], Santos, L. G. [‡], Oliveira, R. G. [§], Afonso, S. P. [¶],
Souza, J. M. [‖], Costa, G. S. N. [**], Lima, F. S. [††], Cotrim, R. M. [‡‡]
Instituto Federal de Educação, Ciência e Tecnologia Baiano - Campus Guanambi, Guanambi, Bahia

http://www.imavs.org/

ABSTRACT

Remotely Piloted Aircraft Systems (RPAS), commonly known as drones, have been integrated into various activities since the first decade of the 21st century, driven by technological advancements and globalization [1]. However, the high cost of drones, particularly due to the image capture device, limits their accessibility in social and educational sectors, with the imaging component representing up to half of the total equipment cost. To address this challenge, this research evaluates the effectiveness of four affordable cameras — IMX519, IPCAM, ESP32, and ANALOGIC — that were not originally designed for drones, aiming to identify their potential in detecting two-dimensional ArUco markers. The analysis considers several criteria, including resolution, power consumption, brightness, weight, and detection performance. The tests were conducted outdoors, with the cameras configured to capture images at five flight altitudes (3, 6, 9, 12, and 15 meters) under varying light conditions throughout the day. ArUco markers of different sizes (100, 200, 300, and 400 mm) were used to assess the cameras' detection capabilities. The results revealed significant variations in performance among the cameras. The IPCAM stood out with the best overall detection average, consistently demonstrating superior performance, particularly at lower altitudes and with 400 mm markers. Although the ESP32 offers advantages in power consumption, weight, and cost, these differences are secondary compared to the IPCAM's remarkable superiority in marker detection. Therefore, the results indicate that the IPCAM is the most recommended choice for efficient ArUco marker detection, offering the best balance between performance and cost-effectiveness for RPAS applications with computer vision systems.

[*]Email address: vitorjo8gbi@gmail.com
[†]Email address: durvaljunior117@gmail.com
[‡]Email address: leandro.santos@ifbaiano.edu.br
[§]Email address: rafaelgomesdeoliveiraa@gmail.com
[¶]Email address: saviopessoaafonso@gmail.com
[‖]Email address: jeovanamiranda218@gmail.com
[**]Email address: gustavosncosta@gmail.com
[††]Email address: fabio.lima@ifbaiano.edu.br
[‡‡]Email address: reinaldo.cotrim@ifbaiano.edu.br

1 INTRODUCTION

The so-called Remotely Piloted Aircraft Systems (RPAS), popularly known as "drones", are devices that have become intrinsically linked to various industrial, operational, social and leisure activities since the first decade of the 21st century. These technological transformations driven by globalization require individuals to adapt to the contemporary world [1].

However, there are still several challenges that prevent the integration of these innovations into the daily lives of society, within low-capital corporations, and in education, particularly due to their high cost.

One of the factors contributing to their elevated price is the image and video capture device, which is essential for several tasks, such as area mapping, object detection, precision agriculture, and inspections. However, in many cases, this device can represent up to half of the total equipment cost. Thus, there is a need for the development or functional repurposing of photography and filming technologies aimed at increasing the accessibility of RPAS.

From this perspective, this research proposes a functional redirection by comparing the effectiveness of four affordable camera models, which, although not originally developed for use in drones, are evaluated in this context to identify their potential application in such devices. The different cameras were tested with the detection of two-dimensional ArUco markers of different sizes, considering variables such as light levels (LUX) and flight altitude, elements that are capable of influencing the system's performance.

According to the analysis of the results obtained, it is possible to identify which environmental conditions and camera combinations offer the best accuracy in detecting ArUco markers and, consequently, determine the device with the greatest cost-benefit for use in RPAS with embedded computer vision systems.

2 RELATED WORKS

ArUco markers are fiducial squares with a black-and-white binary pattern, widely used for pose estimation in several applications, including autonomous robots, virtual trainers, and unmanned aerial vehicles [2] [3] [4]. The OpenCV

http://www.imavs.org/

library, used for developing computer vision algorithms, has a module dedicated to the detection of these markers [5].

ArUco markers are configured in a 7x7 matrix, totaling 49 cells, with 24 allocated to the border and 25 to the data area. This area is subdivided into 15 cells to ensure integrity and 10 for information encoding, allowing for up to 1,024 combinations. Integrity is ensured through a logical verification scheme across the columns, enhancing the robustness of the detection system [6].

Despite the choice of the ArUco marker, other binary square fiducial markers, such as Matrix [7] and ARTag [8], also have similar applications. When detected by cameras, these markers enable the execution of a single pre-established function in the UAV, while also allowing the estimation of the position of a monocular camera at minimal cost, with high robustness and speed.

In a literature review on autonomous vehicles using RGB camera-based vision, [9] highlighted the effectiveness of these cameras under different visibility conditions and in complex scenarios. Studies such as those by [10], [11], and [12] demonstrated that the use of monocular cameras, combined with deep learning, overcame the limitations of other sensors such as LiDAR, especially in detecting objects at different scales, under various lighting conditions, and across different viewing angles and speeds, enabling precise mapping, localization, positioning, and landing of the aircraft.

[3] conducted a study that combined two cameras with distinct characteristics — a fisheye lens camera and a stereoscopic camera — with the objective of facilitating the autonomous landing of drones on mobile ground vehicles in GPS-denied environments. This work is closely related to the present study, as the features of different cameras and their adaptability to various lighting conditions are critical factors for both accurate detection and precise drone control.

A recent study by [13], which utilized an event-based camera for ArUco marker detection, revealed that this type of sensor achieves a significantly higher detection rate under high-speed conditions and varying levels of illumination compared to conventional RGB cameras. Key variables such as flight altitude, lighting conditions, and rapid camera movement were found to influence detection efficiency, highlighting the importance of considering these factors when assessing the performance of different drone-mounted cameras in ArUco marker detection.

3 MATERIALS AND METHODS

Initially, image capture devices were selected based on availability and a maximum configuration cost of $150.00, including the device and the equipment required for its full operation. Based on these criteria, four different configurations were designated for standardization purposes as IMX519, IP-CAM, ESP32, and ANALOGIC.

To perform the comparative analysis, the different systems were tested with the detection of standardized ArUco two-dimensional markers in sizes of 100, 200, 300, and 400 mm using the 'Original ArUco' dictionary, although for the purposes of this analysis, only ID48 was used

Detections were performed in an open-air environment, on the surface of a road built with concrete blocks. ArUco markers of various sizes were arranged side by side for image capture. The four camera configurations were mounted on the RPA to ensure that all images were captured under the same conditions.

Flights were set at altitudes of 3, 6, 9, 12, and 15 meters, and for each established flight altitude, the cameras were configured to capture 30 images, excluding those where one or more of the four marker sizes were out of frame. The collection resulted in a total of 90 images per flight altitude for each camera. Thus, each camera produced 450 images, totaling 1,800 images in the dataset

Since the images were captured outdoors at different times throughout the day, ambient light levels were also measured, as lighting is a determining factor in the quality of the image recorded by the camera sensor. The light was measured using the INS-1381 Digital Luxmeter during each flight conducted for image capture. The luxmeter was leveled in a horizontal position on a pedestal 120 cm above the ground and positioned less than 2 meters from the ArUco markers. Additionally, image metadata, such as longitude, latitude, and time, were recorded.

3.1 Setups

3.1.1 Setup 1: IMX519

Configuration 1 utilizes the IMX519 digital image capture sensor, which provides 16 megapixels (Mp) and supports a maximum resolution of 4656 x 3496 pixels, with a transfer rate of up to 9 frames per second (fps) at this resolution. At lower resolutions, the sensor can achieve up to 120 fps. However, for the purposes of this study, a resolution of 1920 x 1080 pixels at 30 fps was selected.

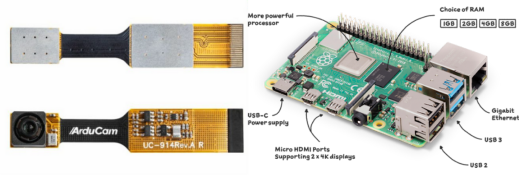


Figure 1: IMX519 camera module and Raspberry Pi 4B that make up the setup 1

It also features an 80° (diagonal) field of view, a 1/2.53-inch CMOS sensor, an f/1.75 lens, dimensions of 34 mm x 34 mm x 26.23 mm, a weight of 4g, and operates within a temperature range of 0ºC to 70ºC with an operating voltage of 5 volts.

For its operation, the Raspberry Pi 4 Model B single-board computer was utilized, connected via a flat cable to the

Camera Serial Interface (CSI). According to the datasheet released by Raspberry Pi (Trading) Ltd. (2019), this model is equipped with a high-performance 64-bit quad-core processor, offering robust performance and thus enabling the efficient execution of complex tasks.

3.1.2 Setup 2: IPCAM

Configuration 2 consists of the Hw0043 image capture device, a security camera with 1 MP and a maximum resolution of 1280 x 720 pixels, which was the resolution used. It supports an image transfer rate of up to 30 fps and can be connected via Ethernet port or WiFi.



Figure 2: Wanscam HW0043 Wi-Fi security camera

The device operates in a temperature range of -10ºC to 50ºC, features a 110º diagonal field of view (FOV), a 1/4" CMOS sensor, a 3.6 mm lens, and operates at a voltage of 5V. To reduce its weight, the protective aluminum casing was removed; therefore, the camera now weighs approximately 31g and measures 32.3 mm x 32.8 mm x 38.9 mm. This camera is integrated with the MT7601un WiFi module, which operates in accordance with the IEEE 802.11b standard and can achieve speeds of up to 150 Mbps.

3.1.3 Setup 3: ESP32

Configuration 3 is composed of the OV2640 IoT image capture device, manufactured by OmniVision. It features a 2-megapixel sensor, a maximum resolution of 1600x1200 pixels, and an image transfer rate of up to 60 fps. However, a resolution of 1024x768 pixels was utilized for this study. The device includes a 1/4-inch CMOS sensor, dimensions of 8.1 mm x 8.1 mm x 5.7 mm, and operates within a temperature range of 0ºC to 50ºC, with a voltage requirement of 5V.



Figure 3: OV2640 62º Camera

Using a small flat cable, the OV2640 camera is integrated into the ESP32-CAM board, which includes a serial input compatible with the device. This assembly weighs 10 grams, with a lens that has a 62º field of view (FOV), operates at 5V, and exhibits a power consumption ranging between 160 mA and 310 mA during operation.

3.1.4 Setup 4: ANALOGIC

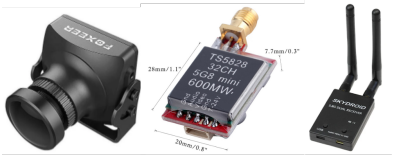


Figure 4: Foxeer camera, video transmitter and receiver that make up the setup 4

The Foxeer Monster V2 camera is the only analog device tested. It offers a resolution of 1280 x 960, with a 1/2.9-inch CMOS sensor and a 3.6 mm lens. Its operating voltage is 5V, and it functions within a temperature range of -10ºC to 50ºC. Unlike the others, the Foxeer Monster V2 has 1200 TVL (Television Lines). To operate this camera, it is necessary to integrate a video transmitter system into the aircraft, specifically using the TS5828 model with 32 channels and 600 mW. The ground station, on the other hand, utilized a Skydroid UVC 5.8 video receiver, connected to a computer via a USB port. The camera has dimensions of 28 mm x 26 mm x 30 mm and, together with the video transmitter, weighs 19 grams.

3.2 RPA Specifications

The aircraft used for this study was built on a Tarot 650 frame, configured as a quadcopter multirotor with a carbon fiber structure. It was equipped with 30A Blheli Oneshot-125 LittleBee Electronic Speed Controllers (ESCs) and Sunnysky 3508-600 KV brushless motors. Additionally, a PixHawk 1 (3DR) flight controller, configured with Arducopter firmware, was utilized to maintain the control and stability of the aircraft, enabling automated flights.

A GPS module integrated with a compass was used to determine latitude and longitude, allowing for precise identification of the quadcopter's orientation. Similarly, to accurately measure the drone's altitude, the Benewake TFmini Plus module, equipped with a laser distance sensor, was employed, providing measurements up to 6 meters with an error margin of ±5 cm.

3.3 Measuring electrical consumption

The electrical consumption tests of the four setups were carried out on a bench, using the Riden DC6012 source, which provides instantaneous data on voltage, current, power, and consumption of the equipment connected to it.

http://www.imavs.org/

3.4 *ArUco marker recognition*

The objective of the ArUco marker recognition test is to determine how the distance between markers and ambient lighting affects each camera setup, thereby analyzing each setup's ability to identify reference points of varying sizes.

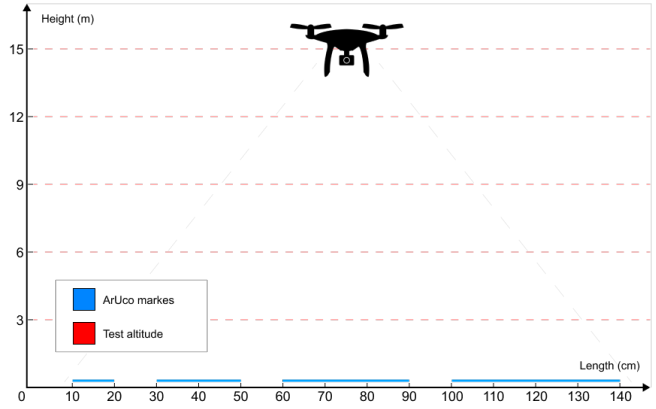


Figure 5: Representative illustration of the positioning of arUco markers during flights to capture images at different heights.

According to [14], system evaluations can be qualitative or quantitative. Qualitative evaluation involves obtaining feedback from users about their satisfaction with the system, while quantitative evaluation objectively measures system performance and is often applied in detection and classification algorithms. [15] highlights the confusion matrix as an essential tool for evaluating binary classification, allowing the observation of correct and incorrect predictions, categorized as True Positive (TP), False Negative (FN), False Positive (FP), and True Negative (TN). Varoquaux and Colliot (2023) and Markoulidakis et al. (2021) illustrate the importance of this matrix for calculating evaluation metrics, where Precision = TP / (TP + FP) represents the proportion of correctly classified positives, and Accuracy = (TP + TN) / (TP + FN + FP + TN) reflects the overall proportion of correctly classified instances.

Therefore, the values of each image were classified into a confusion matrix to analyze how these variables affect detection based on precision and accuracy metrics.

3.5 *Algorithm*

TTo optimize the testing process and obtain more accurate data, software was developed that directly communicates with the drone and the selected cameras. This software automates drone control and ArUco detection, in addition to saving the processed images. This enables a precise evaluation of the performance of the image capture devices used, significantly reducing the time and effort required compared to manual methods.

The development of this software was carried out using the object-oriented programming language Python, version 3.10.9. Two libraries were essential for automating the tests: Pymavlink, version 2.4.41, which manages drone communication and control via the MAVLink protocol, and OpenCV, version 4.9.0.80, which facilitates communication with the cameras and ArUco detection through digital image processing.

The code employs the object-oriented paradigm, creating classes that encapsulate behavior and preserve their internal state through attributes and methods. A specific class was developed for each logical or physical component of the test. For example, the 'Drone' class communicates directly with the physical drone and controls it through its methods. Each camera was also abstracted as a class.

The Abstract Factory design pattern was used to create classes for each camera, using inheritance to maintain their common characteristics and polymorphism to expose differing behaviors. These approaches ensure that the existing code does not need to be modified, following the Open-Closed and Dependency Inversion principles. This ensures modular and maintainable code, facilitating the addition of new devices and functionalities in the future

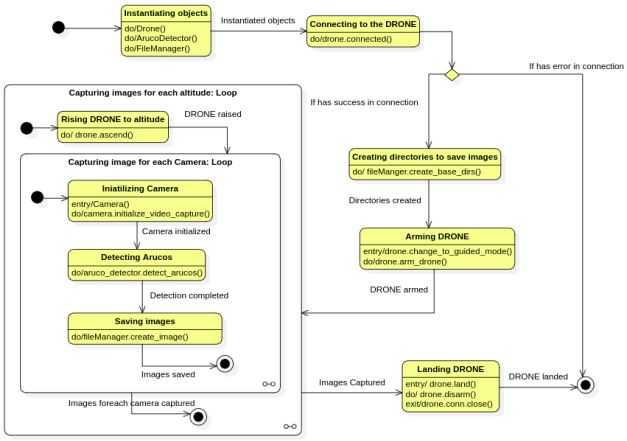


Figure 6: State machine diagram illustrating the algorithm's states during testing

In the state machine diagram shown in Figure 6, the software states during the execution of the tests can be observed. The process begins with the instantiation of the objects necessary to perform the tasks, followed by an attempt to connect to the drone via telemetry. If the connection is not established, the code is terminated. If the connection is successful, directories are created to save the images captured during the test.

Next, the drone's flight mode is set to GUIDED, where the drone is controlled by commands sent by an operator or software, rather than following a pre-programmed flight plan or being manually controlled, and then its motors are armed. For each altitude defined in the test, the drone ascends and detects the ArUcos in all configured cameras. After complet-

http://www.imavs.org/

ing the entire process, the drone switches to LAND mode and lands, finally closing the connection between the software and the drone.

4 RESULTS AND DISCUSSIONS

Table 1 presents the data on electrical consumption and weight of the configurations evaluated in this study. The electrical consumption and weight of the camera directly impact flight autonomy and are, therefore, critical factors to be considered when defining the drone configuration.

Setup	CAM	V (V)	C (Ah)	W (g)	COST (US$)
1	IMX519 + Raspberry PI 4B.	5.0	0.40	50	144.00
2	Wanscam Hw0043 + Wifi MT7601UN.	5.0	0.20	31	25.00
3	OV2640 IOT + ESP32-Cam.	5.0	0.16	10	6.00
4	Foxeer Monster V2 + TS5828.	12.0	0.45	19	60.00

Table 1: Specifications and consumption of the setups studied
CAM = camera, V = voltage (V), C = current (Ah), W = weight (g).

Configuration 1 has the heaviest weight (50 g) and relatively high power consumption (0.40 Ah) compared to the others. The combination of its weight and power consumption results in a significant reduction in the drone's flight time relative to the other configurations. Configuration 2, weighing 31 g and consuming 0.20 Ah, offers a good balance between weight and power consumption, which results in greater flight autonomy compared to configurations 1 and 4.

Configuration 3 is the lightest (10 g) and has the lowest power consumption (0.16 Ah), providing the drone with greater autonomy than the other configurations. Configuration 4 has an intermediate weight (19 g) but the highest power consumption (0.45 Ah), leading to reduced flight autonomy.

The analysis of the cameras' performance in detecting ArUco markers reveals that accuracy, a metric used for evaluating classifiers, is influenced by both the marker size and the camera's altitude, as shown in Figure 7.

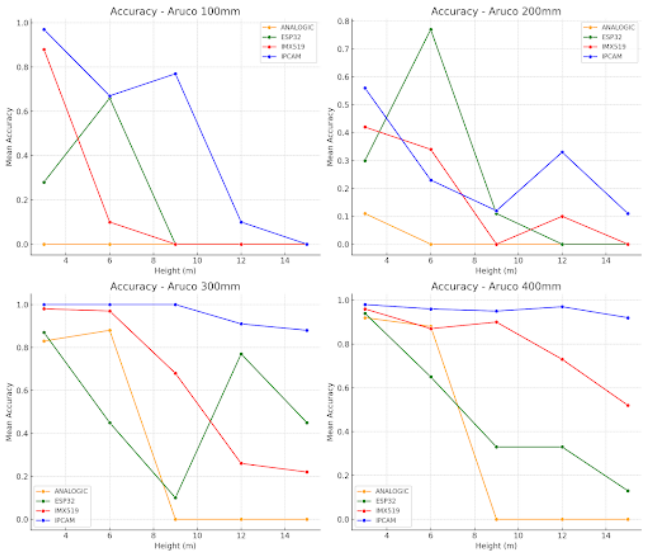


Figure 7: Accuracy of the studied cameras in detecting ArUco markers of different sizes and flight heights

The camera analyses revealed marked differences in their performance depending on the variables tested. The ANALOGIC camera had the lowest accuracy rate among the options evaluated. This camera showed an accuracy rate of almost zero for 100 mm and 200 mm ArUcos at all tested altitudes, indicating significant difficulty in identifying these markers at different heights. However, for 300 mm and 400 mm ArUcos, the ANALOGIC demonstrated better performance at altitudes ranging from 3 to 9 meters. Despite this, its performance dropped drastically above these altitudes, evidencing a limitation in the ability to maintain accuracy at greater distances.

The ESP32 demonstrated a superior ability to handle larger markers, such as 300 mm and 400 mm, even at higher altitudes. This suggests that the ESP32 has good resolution and sensitivity to identify large markers at greater distances. However, the camera consistently performed poorly in detecting smaller ArUcos (100 mm and 200 mm), regardless of altitude. Still, compared to the IMX519, the ESP32 was more effective at lower altitudes for small markers. However, as altitude increased, the IMX519 proved to be more suitable, especially for 300 mm and 400 mm markers, with a higher accuracy rate at greater heights.

The IMX519 showed intermediate performance, but its effectiveness increased with altitude and marker size. It also excelled, especially at higher altitudes with large markers, demonstrating a superior ability to maintain accuracy under these conditions. This suggests that the camera is more adaptable to variations in altitude and marker size, offering a good solution for scenarios where marker height and size are greater.

Finally, the IPCAM achieved the highest accuracy rates among all the cameras tested and proved to be the most sta-

http://www.imavs.org/

http://www.imavs.org/

ble. The IPCAM maintained a high level of consistent performance across different altitudes and ArUco sizes, making it the most reliable choice for practical applications.

Initially, for the 100 mm ArUco marker, a decrease in detection accuracy is observed as altitude increases. The IPCAM maintains an accuracy of 0.68 up to 6 meters, but the graph indicates a sharp decline from 9 meters onward, reaching nearly zero at 12 meters. Despite this, the IPCAM still outperforms the other cameras. The IMX519 showed a detection accuracy of 0.89 only at 3 meters, drastically reducing to near zero at 6 meters, and above this height, detection is null. The ESP32, however, demonstrates an accuracy of 0.30 at 3 meters and reduces to zero from 9 meters, thus maintaining low accuracy from the lowest height. The ANALOGIC camera exhibits the worst performance, with null detections at all altitudes.

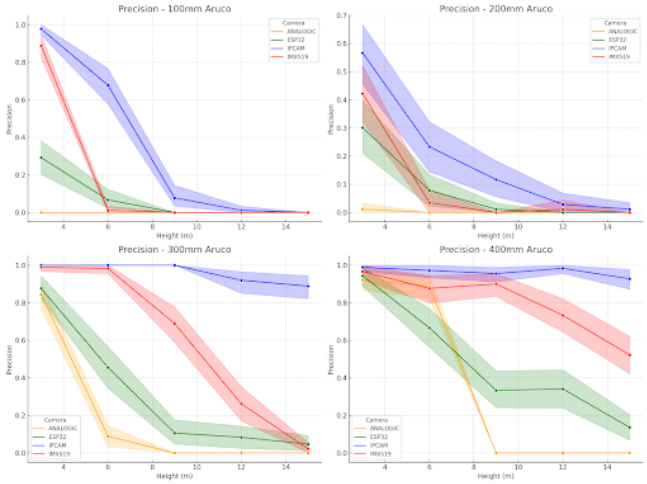


Figure 8: Accuracy in detecting ArUco markers at different heights and sizes for the different cameras studied

When increasing the size of the ArUco marker to 200 mm, the detection accuracy of the ANALOGIC camera remained the poorest performer, with zero detections at all flight altitudes. However, the IPCAM and IMX519 cameras achieved precisions of 0.57 and 0.42, respectively, in detecting the marker at a 3 m altitude, dropping drastically to zero from 9 m onwards. The ANALOGIC camera recorded the lowest precision at all altitudes for this marker size (200 mm). Similarly unsatisfactory, the ESP32 presented a maximum precision of 0.30 at a 3 m altitude, with a sharp decline at 6 m. For this marker size, at altitudes above 6 m, all cameras exhibited poor performance, with precision practically reduced to zero.

Furthermore, for the 300 mm ArUco marker, all cameras demonstrated significant improvement in performance, with precision greater than 0.84 at a 3 m altitude. For this marker size, the IPCAM stood out by maintaining accuracy above 0.88 even at an altitude of 15 m. The IMX519 camera maintained accuracy around 0.98 up to 6 m, decreasing to 0.69 at 9 m. The ESP32 camera achieved the highest accuracy (0.87) at a 3 m altitude, but this dropped sharply to 0.11 at 9 m. As for the ANALOGIC camera, despite achieving an accuracy of 0.84 at 3 m, it failed to detect the marker from 9 m onwards.

Finally, with the 400 mm ArUco marker, the detection rate was the highest across all cameras. The IPCAM camera achieved accuracy above 0.92 even at the highest altitude. Similarly, the IMX519 camera also performed well, maintaining accuracy above 0.73 at a height of 12 m. The ESP32 camera achieved accuracies of 0.94 and 0.67 at heights of 3 and 6 m, respectively, but its performance decreased to 0.33 at heights above 9 m. The ANALOGIC camera, despite achieving accuracies of 0.92 and 0.89 at 3 and 6 m, respectively, did not detect any markers at heights above 9 m, even though it was evaluating the largest marker size in this study.

Figure 9 illustrates the relationship between the True Positive Rate (TPR) and luminosity (in LUX) for different ArUco marker sizes (100 mm, 200 mm, 300 mm, and 400 mm) and various camera types (ANALOGIC, ESP32, IMX519, IPCAM). The TPR is a metric that indicates the proportion of markers correctly identified by the camera relative to the total number of markers present. Consequently, the graph enables a comparative analysis of the cameras' performance under varying lighting conditions and for different marker sizes.

For the 100 mm ArUco markers, all cameras show relatively low TPR rates, especially under low-light conditions. As luminosity increases, performance improves slightly, but detection remains limited for all cameras. Some variations are observed, with the IPCAM and IMX519 cameras achieving slightly higher rates in environments with luminosity levels above 100,000 LUX, although rates remain generally low.

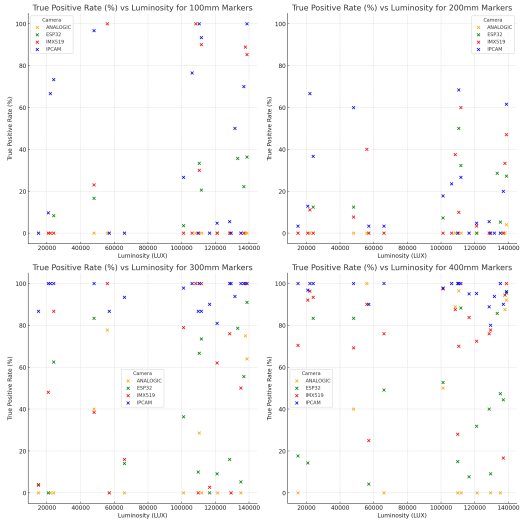


Figure 9: True positive rate in the detection of ArUco markers of different sizes by the cameras studied as a function of brightness.

In the case of the 200 mm markers, there is a similar be-

havior, where TPR rates remain low under low light conditions. As the light intensity increases, especially above 100,000 LUX, the IMX519 and IPCAM cameras begin to exhibit better performance, with higher TPR compared with the other cameras. The ESP32, however, consistently shows low performance under all lighting conditions.

When analyzing the 300 mm markers, an overall improvement in camera performance is observed. TPR rates increase significantly, especially for the IMX519 and IPCAM cameras, which exceed 80% under high light conditions. The ANALOGIC camera also demonstrates reasonable performance, while the ESP32 continues to show lower TPR rates, indicating less efficient performance in detecting these markers.

For 400 mm markers, the IMX519 and IPCAM cameras achieve even higher TPR rates, often close to 100%, under high luminosity conditions. The ANALOGIC camera also maintains solid performance in this scenario. However, the ESP32 shows lower TPR rates compared to the other cameras, even with greater luminosity and larger markers.

5 CONCLUSION

Using the collected data, the relationship between the detection of ArUco markers, observation height, and marker dimensions was analyzed. Generally, the larger the ArUco marker, the higher the detection rate. Conversely, increasing the distance between the camera and the marker causes a significant drop in detection efficiency. Additionally, lighting has proven to be an important factor, with greater light intensity substantially enhancing camera performance, allowing for more accurate identification of the markers.

In this study, the IPCAM camera demonstrated the best overall performance in detecting ArUco markers, particularly at low altitudes and with 400 mm markers. Moreover, it showed significant superiority over the other cameras in detecting smaller markers. When considering other variables, such as power consumption, weight, and acquisition cost, the ESP32 camera exhibited superior performance in all these aspects. However, the differences in these criteria are minimal and do not have a significant impact compared to the more substantial difference in detection performance between the cameras, which clearly favors the IPCAM.

Thus, the results suggest that, among the cameras studied, the IPCAM emerged as the most suitable choice for detecting ArUco markers. These findings were crucial in guiding the Drones Guanambi team in selecting the camera to be used in the indoor and outdoor events of IMAV 2024.

The IMX519 camera, because it needs the Raspberry, is the heaviest and has the second highest power consumption. The Wanscam, on the other hand, had the second lowest power consumption and the highest detection rate of ArUco markers.

When comparing the two cameras, it is crucial to consider other factors. The IMX519 requires connection to a microcomputer, increasing energy consumption, and the need to monitor the status of the microcomputer to ensure correct data transfer. In contrast, the Wanscam Hw0043 Camera offers a simpler connection and data transfer, consumes less energy, and is more affordable.

The IMX519 camera has a higher resolution, reaching up to 16 MP (megapixels). This can be advantageous for teams with more programming experience, allowing programmatic adjustments to the camera's quality and focus, potentially improving the detection of ArUcos.

The choice between the Wanscam Hw0043 Camera and the IMX519 will depend on the specific needs of the project. The Wanscam Hw0043 is a robust and economical choice for intermediate altitudes, while the IMX519, with its high resolution and potential for programmatic adjustments, can offer significant advantages for projects that demand superior image quality and greater technical flexibility, even if it has higher electrical consumption and financial costs.

REFERENCES

[1] I. O. Silva, J. E. B. Rosab, E. L. Hardoimc, and G. Guarim Neto. Educação científica empregando o método steam e um makerspace a partir de uma aula-passeio. *Latin American Journal of Science Education*, 4:22034, 2017.

[2] J. Bacik, F. Durovsky, P. Fedor, and D. Perdukova. Autonomous flying with quadrocopter using fuzzy control and aruco markers. *Intelligent Service Robotics*, 10(2):185–194, 2017.

[3] Tao Yang, Qiang Ren, Fangbing Zhang, Bolin Xie, Hailei Ren, Jing Li, and Yanning Zhang. Hybrid camera array-based uav auto-landing on moving ugv in gps-denied environment. *Remote Sensing*, 10(11):1829, 2018. Accessed: 29 Aug. 2024.

[4] Jakob Grimm Hansen and Rui Pimentel de Figueiredo. Active object detection and tracking using gimbal mechanisms for autonomous drone applications. *Drones*, 8(55):1–15, 2024. Accessed on 30 August 2024.

[5] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer. Aruco: Augmented reality library based on opencv. https://sourceforge.net/projects/aruco/, 2019. Accessed on 30 June 2024.

[6] S. L. A. Silva, A. M. G. Tommaselli, and A. O. Artero. Evaluation of the usability of aruco markers in robotic applications. *Brazilian Journal of Biosystems Engineering*, 8(3):198–207, 2014. Accessed on: Aug. 30, 2024.

[7] Jun Rekimoto. Matrix: A realtime object identification and registration method for augmented reality. In *Pro-*

http://www.imavs.org/

ceedings of the 3rd Asia Pacific Conference on Computer Human Interaction, pages 63–68, July 1998.

[8] Mark Fiala. Designing highly reliable fiducial markers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(7):1317–1324, Jul. 2010.

[9] M. I. Pavel, S. Y. Tan, and A. Abdullah. Vision-based autonomous vehicle systems based on deep learning: A systematic literature review. *Applied Sciences*, 12(14):6831, 2022. Accessed on 30 August 2024.

[10] Alexandre Borowczyk, Duc-Tien Nguyen, André Phu-Van Nguyen, Dang Quang Nguyen, David Saussié, and Jerome Le Ny. Autonomous landing of a quadcopter on a high-speed ground vehicle. *Journal of Guidance, Control, and Dynamics*, 40(4):1037–1049, 2017.

[11] Rafael Muñoz-Salinas, Manuel J. Marín-Jiménez, Enrique Yeguas-Bolívar, and R. Medina-Carnicer. Mapping and localization from planar markers. *Pattern Recognition*, 2017. Preprint available at arXiv:1606.00151v2.

[12] Xiaowei Hu, Xuemiao Xu, Yongjie Xiao, Hao Chen, Shengfeng He, Jing Qin, and Pheng-Ann Heng. Sinet: A scale-insensitive convolutional neural network for fast vehicle detection. *IEEE Transactions on Intelligent Transportation Systems*, 20(3):1010–1019, March 2019.

[13] Hamid Sarmadi, Rafael Muñoz-Salinas, Miguel A. Olivares-Mendez, and Rafael Medina-Carnicer. Detection of binary square fiducial markers using an event camera. *IEEE Access*, 9:27813–27825, 2021. Accessed: 29 August 2024.

[14] Hercules Dalianis. Evaluation metrics and evaluation. In *Clinical Text Mining: Secondary Use of Electronic Patient Records*, pages 45–53. Springer International Publishing, Cham, 2018.

[15] David M W Powers. Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation. *Machine Learning Technologies*, 2, January 2008.

http://www.imavs.org/

Photovoltaic plant monitoring and inspection through synergic integration of UAVs and IoT

Javier Melero-Deza*, Rafael Perez-Segui*, Pedro Arias-Perez*, Miguel Fernandez-Cortizas*, David Perez-Saura*, Guillermo GP-Lenza*, Miguel Tradacete-Ágreda†, Carlos Santos Pérez‡, Francisco Javier Rodríguez Sánchez† and Pascual Campoy*

http://www.imavs.org/

ABSTRACT

Photovoltaic (PV) panels play a crucial role in renewable energy generation. Ensuring their optimal performance and longevity requires regular inspection. Traditional methods rely on scheduled inspections with manual, offline path planning. Our system employs a dynamic online planning algorithm that allows for real-time task allocation and inspection on a per-panel basis. In this paper, we propose a new approach where each panel is embedded with IoT sensors that communicate inspection requests to a sensored UAV swarm. This allows to create a specific inspection plan for each UAV, adapting to panel conditions and requests. Our approach contributes to the advancement of sustainable energy systems by focusing on individual panels when demanded, reducing overall inspection time and enhancing accuracy. We present the design, implementation, and evaluation of this system. [1]

1 INTRODUCTION

The transition to a sustainable energy future necessitates innovative solutions to maximize the efficiency and effectiveness of renewable energy sources. In this context, solar photovoltaics (PV) has been the fastest growing power generation technology in the world over the last decade [1]. All the scenarios towards a climate neutral energy assign a central role to PV. The widespread adoption of photovoltaic installations holds immense promise, but also presents unique challenges. To fully unlock the potential of solar energy, it is crucial to optimize the economic and energy performance of PV plants while ensuring seamless integration with existing grid infrastructure.

Another major challenge is how to carry out the monitoring and maintenance of photovoltaic plants. Many of these plants are too large, making their inspection very time-consuming if it has to be done by workers. To address this issue, various proposals have emerged, among which the use of drone fleets offers an efficient and novel solution [2]. To accomplish this mission, drones are equipped with cameras and various types of instruments [3], as shown in Figure 1. While thermal cameras mounted on UAVs can detect most PV issues [4], additional PV mounted sensing is needed, as not every issue in PV panels manifests itself in form of temperature increase, depicted in Figure 2. Additionally, it is worth noting that early fault detection is crucial in the maintenance of photovoltaic plants to ensure adequate performance and prevent defects from spreading to healthy areas. Furthermore, detecting the different aging mechanisms that can also affect photovoltaic modules is important, as a mismatch phenomenon can occur between them [5].



Figure 1: DJI M300 inspecting a PV plant.

The primary objective of this work is to integrate state-of-the-art communication technologies and autonomous UAV capabilities to streamline the control and monitoring of PV plants.

To do that, we use autonomous swarm planning algorithm for drone fleets tailored specifically for solar park inspection. These plans are generated by an online planning system that uses information coming from low-cost IoT systems for digitalized solar panels capable of self-assessment and communication in order to ensure efficient and comprehensive monitoring of PV plants.

*Computer Vision and Aerial Robotics Group at Universidad Politécnica de Madrid, Spain (CVAR-UPM) at the Centre for Automation and Robotics C.A.R. (UPM-CSIC). Email address(es): javier.mdeza@upm.es

†Department of Electronics, University of Alcalá, Alcalá de Henares, Madrid, Spain

‡Department of Signal Theory and Communications, University of Alcalá, Alcalá de Henares, Madrid, Spain

[1]System validation video

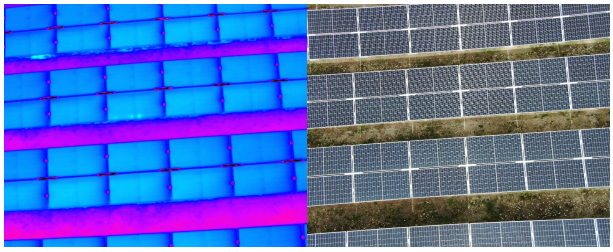


Figure 2: Image capture during the inspection process, with the thermal and RGB images.

This planning approach is swarm-oriented, meaning that each generated plan takes into account specific factors to determine which drone is assigned to which plan, along with the corresponding parameters.

We have also integrated computer vision and deep learning algorithms that work with the imaging sensors mounted on the drones to perform real time detection and evaluation of rgb and thermal images.

By pursuing this objective, we aspire to contribute to the advancement of solar energy technologies and facilitate the transition towards a more sustainable energy future. Our findings not only hold the potential to enhance the performance and reliability of PV installations but also play a crucial role in mitigating the effects of global warming by maximizing the utilization of renewable energy sources.

Concisely, the main contributions of this work are as follows:

1. Design of a novel IoT architecture that integrates solar panels and UAV swarm in a PV plant.
2. An innovative swarm-oriented online-planning algorithm to optimize on demand PV inspections.
3. An onboard real-time visual inspection and evaluation pipeline for rgb and thermal images.
4. Experimental evaluation of the system in a real photovoltaic plant, providing precise and efficient inspections.

2 INDUSTRY DRONE-BASED INSPECTION TECHNIQUES AND PV MONITORING

The rapid expansion of PV markets has spurred the development of large-scale solar power plants, intensifying the need for more advanced inspection and monitoring tools. Traditionally, manual inspections have been the norm [6], but there's a growing shift towards employing more dynamic systems, particularly drones, for these tasks. Drones are increasingly recognized as a fitting solution for the solar industry due to their diverse surveillance and monitoring capabilities, long-range inspection potential, and ease of control. Over recent years, they have gained popularity for their ability to swiftly monitor expansive solar parks more efficiently than human inspection teams. Equipped with sensing technology, drones efficiently gather essential data and transmit it to the cloud for analysis, significantly reducing time and enhancing accuracy.

Current state of the art usage of drone-based PV plant monitoring suggests having an experienced drone pilot to remotely manipulate the drone [7]. However, for a fully autonomous drone, the focus and range (i.e., correct temperature range set on thermal camera) of the cameras, as well as the flying distance are difficult to dynamically reconfigure, and they require proper settings during image capture.

During an automated aerial infrared thermography operation, the UAV navigates through a series of waypoints, ensuring comprehensive coverage of the solar plant's modules [8]. Consequently, the deployment of a refined path-planning algorithm is crucial to achieve the most efficient route in terms of time and battery usage [9].

Moving on to fully autonomous UAVs, we can clearly differentiate between a planning phase and a navigation phase.

When focusing only on planning of UAVs, a semi-supervised remote control system is proposed in [10], where there is no planning needed as it is integrated in the human pilot control loop. In [11] a GUI is integrated with the capacity of defining waypoints for more specific curvilinear terrain shapes or polygons in case the layout allows it, to then create a grid shaped path within that polygon. A sequence of georeferenced waypoints is introduced in the system prior to navigation phase in [12], which implies user having to look up to exact GPS coordinates with a third party tool. A form of semi-automated planning is performed in [13] in which the boundary information can be gained from design drawings of the photovoltaic plant, GPS, and aerial images together with Geographic Information Systems. A polygon is then built and path is created using back and forth algorithm. A previously designed trajectory is used in [14], being unspecified the way of creating it. In [15], travelling salesman shortest path algorithm to devise a route encompassing a random assortment of modules that collectively depict the entire photovoltaic (PV) plant, thus optimizing battery utilization is utilized. In [16] a strategy is developed utilizing density clustering, boustrophedon motion planning, and Bezier curves. Similarly, path planning optimization algorithm using Bezier curves is crafted in [17], in conjunction with particle swarm optimization. This approach also factors in the dynamics of flight attitude, the constraints of the gimbal, and the overall path distance.

All of previously mentioned autonomous or semi-autonomous UAVs PV inspection systems have one thing in common, planning is made offline, which means there would be a limitation in time and workload if the inspection zone layout changes and planning needs to be reconfigured.

Motivated by this scarcity, we present a solution that addresses the problematic of offline planning by enabling specific panel and drone communication in order to perform online per-panel planning.

http://www.imavs.org/

http://www.imavs.org/

3 AUTOMATED ONLINE INSPECTION SYSTEM

The main problem addressed in this work is the implementation of an on-demand panel inspection system that is able to integrate PV intelligent monitoring system with an aerial inspection system.

3.1 IoT Architecture Overview

The pipeline of the automated online inspection is composed by three main components, depicted in Figure 3. Alongside, we propose a communication scheme to connect these components. Moreover, we develop a two-step swarm-oriented planning strategy, approach and inspection, to adapt to on-call PV panel inspections. The three main components of the system are:

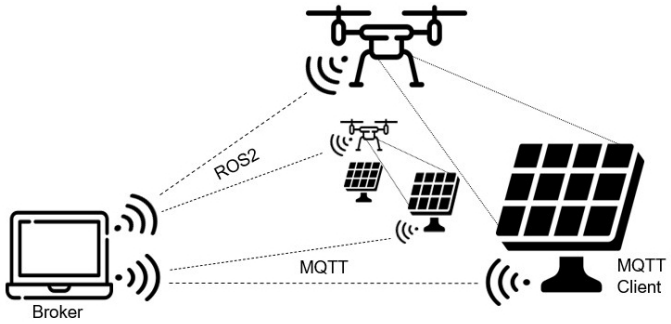


Figure 3: Overview of the inspection system components. It is composed by the PV plant intelligent monitoring subsystem, with the IoT integrated within every panel, the aerial inspection subsystem, with a swarm of UAVs, and a ground station, acting as a manager.

- **Intelligent PV plant monitoring subsystem**. The intelligent PV plant monitoring is composed by IoT boards that have been wired to the electrical system of the panels, along with self-diagnosing tools capable of generating different types of alerts. These alerts reach to the ground station in form of request. Additionally, aerial inspection results are sent back from the ground station to the panels as responses in form of hot-spot location within the panel.
- **Ground station**. The ground station serves as central node, responsible for processing the information received from the panels and managing the aerial inspection subsystem. It acts as a broker between both subsystems, assigning PV panel request to UAVs in the swarm and monitoring inspections in progress.
- **Aerial inspection subsystem**. The aerial inspection system consists of a swarm of UAVs capable of capturing and processing thermal images of the PV panels. The swarm is homogeneous, carrying all UAVs the same payload formed by a gimbal and an hybrid RGB-IR camera.

3.2 Communication Scheme

The communication scheme is responsible for connecting the PV plant's intelligent monitoring subsystem with the aerial inspection subsystem through the ground station.

To achieve this, two communication protocols are used. On one hand, the communication between the PV plant and the ground station is carried out using the MQTT protocol. On the other hand, the connection between the ground station and the agents of the aerial swarm is established using ROS 2, as depicted in Figure 4.

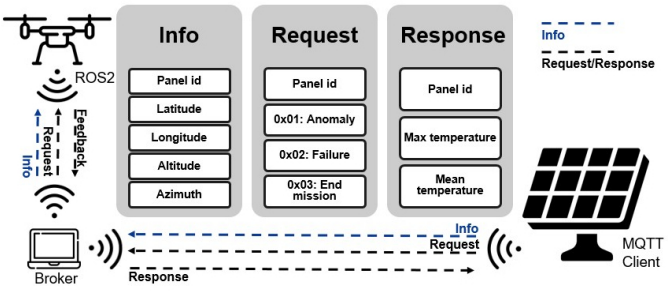


Figure 4: Communication scheme layout. Blue dashed lines stands for *info* interface, while black dashed lines for *request/response* interfaces. Boxes on top show fields of each message for every communication interface.

We establish a link between each PV panel and each UAV using three different types of message interfaces: *info*, *request*, and *response*. Table 1 summarizes these interfaces on both sides, PV plant and UAVs systems.

Interface	Direction	Trigger Event
Info	PV $\rightarrow$ UAVs	Panel initialize
Request	PV $\rightarrow$ UAVs	Panel failure
Response	UAVs $\rightarrow$ PV	Image processed

Table 1: The defined communication interfaces between the intelligent PV plant monitoring system and the aerial inspection system, along with their respective trigger events.

In the MQTT side, the *info* interface is used to send positioning information of each PV panel that is needed for the ground station to build a custom plan for that panel. The *request* interface is utilized by the panel to request an inspection when an alarm is triggered by the self-diagnosing tool of the IoT board connected to the panel sensors. The *response* interface is employed to receive the requested inspection results taken by the aerial subsystem, mainly the maximum, minimum and mean measured temperature.

In the ROS 2 side, the *info* interface is utilized to localize the positions of the PV panels relative to the swarm coordinate system. The *request* interface is employed to assign and define the mission that the UAV must perform to inspect the panel. The *response* interface is used to transmit the tem-

perature information of the panel, which results from image processing by the UAV.

3.3 Swarm-oriented Online Planning Algorithm

The planning algorithm is in charge of translating from request to missions, assigned to UAVs. These inspection missions are managed by the ground station. For every request sent, the ground station assigns a new mission to a specific UAV, and then monitors its inspection. The UAV assignment is based on the distance to the panel position where the request came from, aiming to optimize flight time and, consequently, energy consumption. When the mission reaches the UAV, it is added to its inner First In First Out (FIFO) queue. The UAV will perform each inspection in the queue sequentially, and once the queue is empty, it will land in the safety zone waiting for new inspection requests.

The ground station defines two phases of the mission: approaching to the panel and performing the inspection.

1. **Approach phase:** The approach maneuver is done at a different height for each UAV in flight, and different from the inspection height, to avoid collisions in case of trajectory crossing. If the UAV starts from the ground, the maneuver will require a prior takeoff to the specified height, and if it is already in the air, it will perform a vertical movement only. The approach point is located at a distance from the perpendicular to the panel, using the position and elevation information of the panel.

2. **Inspection phase:** The goal of this phase is to capture a image of the panel and obtain its temperature data. To achieve this, initially, the UAV positions itself at the relative image capture distance from the panel. Subsequently, it adjusts the gimbal to center the panel in the image. Next, the image is captured and processed, with the results being sent to the ground station. Finally, the UAV returns to the approach point, with the security height, awaiting a new request of this panel inspection, or concluding it to proceed with the next task in its queue.

We have used the behavior-based interface of Aerostack2 [18] for mission definition. A mission is formed by tasks, and each task correspond to a behavior from Aerostack2. Figure 5 shows a description of an inspection mission when a panel makes a request, while a visual example of the generated mission is depicted in Figure 6.

3.4 Onboard Real-Time Visual Inspection

During the aerial inspection phase, there are two key points in our strategy to extract the temperature data from the panel. First, centering the panel in the image, and second a three-step onboard image processing pipeline.

The camera gimbal is used for centering the panel in the image, which is important for a better analysis later. To

```
{"takeoff": {"z": approach_height, "frame_id": "earth"}},
{"go_to": {"position": [current_pos_x, current_pos_y,
    approach_height], "frame_id": "earth"}},
{"go_to": {"position": [0.0, 0.0, approach_height], "
    frame_id": "panel_x"}}
```

```
{"go_to": {"position": [0.0, 0.0, capture_distance], "
    frame_id": "panel_x"}},
{"gimbal": {"frame_id": "panel_x"}},
{"image_process": {}},
{"go_to": {"position": [0.0, 0.0, approach_height], "
    frame_id": "panel_x"}}
```

Figure 5: Approach (top) and inspection (bottom) of a panel inspection mission definition using behavior-based tasks from the Aerostack2 standard.

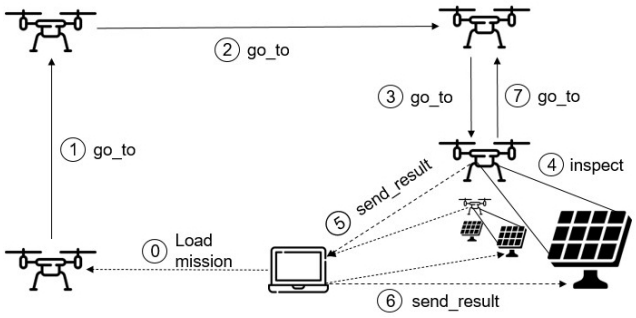


Figure 6: Planned mission relative to PV panel: step 0, 1 and 2 constitute approaching phase, being step 1 executed only if the UAV is landed first. Inspection phase are formed by steps 3 to 7, both included.

achieve this, we introduce a multi-frame coordinate system, as shown in Figure 7. The variables to be controlled are the translation of the UAV with respect to the Earth ($\mathbf{T}^{\text{UAV}}_{\text{earth}}$) and the rotation of the gimbal with respect to the UAV ($\mathbf{R}^{\text{camera}}_{\text{UAV}}$). The desired reference is the camera's translation ($\mathbf{T}^{\text{camera}}_{\text{panel}}$), defined by the desired image capture distance, and its orientation ($\mathbf{R}^{\text{camera}}_{\text{panel}}$), which should be perpendicular to the panel, as shown in Equation 1. The UAV's frame positioning uses its localization system, while the panel's frame positioning uses the information received from the panel.

$$\begin{aligned} \mathbf{T}^{\text{UAV}}_{\text{earth}} &= \mathbf{T}^{\text{panel}}_{\text{earth}} \cdot \mathbf{T}^{\text{camera}}_{\text{panel}} \cdot \mathbf{T}^{\text{UAV}}_{\text{camera}} \\ \mathbf{R}^{\text{camera}}_{\text{UAV}} &= \mathbf{R}^{\text{earth}}_{\text{UAV}} \cdot \mathbf{R}^{\text{panel}}_{\text{earth}} \cdot \mathbf{R}^{\text{camera}}_{\text{panel}} \end{aligned} \tag{1}$$

Once the PV panel to capture is centered in the image, a picture is shot and the onboard image processing pipeline starts. The pipeline's objective is to measure the maximum, average, and minimum temperature of the panel, for transmission to the intelligent monitoring subsystem. Our approach is optimized for embedded platforms and to fulfill real-time capabilities. To achieve this, we propose a three-step strategy. First, we perform a keypoint-based (vertices) panel detection using the framework described in [19]. Subsequently, the

http://www.imavs.org/

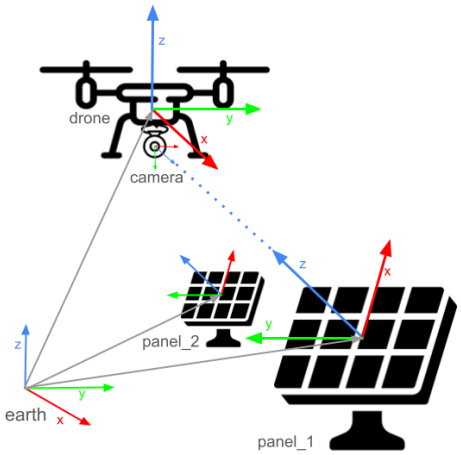


Figure 7: Multi-frame coordinate system used for calculating the positioning of the UAV relative to the panel and the orientation of the camera relative to the UAV.

central panel of the image is segmented using the vertices previously detected. Last, the thermal image is processed finding local maxima and minima over the temperature values of the segmented panel, Figure 8 shows the described image processing pipeline.

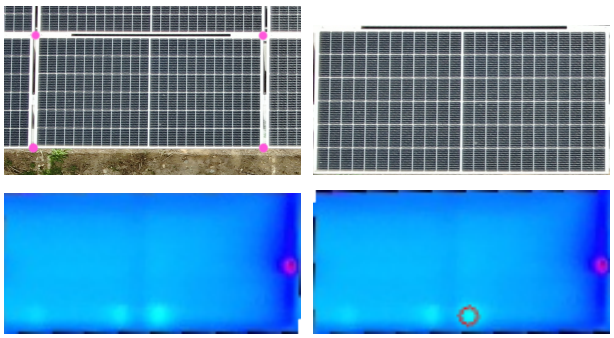


Figure 8: Sequence of the image processing pipeline, from right to left and top to bottom; panel vertices detection, panel segmentation and local maximum detected in thermal image.

4 EXPERIMENTAL VALIDATION

In order to validate our inspection strategy, we performed extensive experiments on simulation and real world environments. We first followed a simulation to real methodology in order to validate our system, then we performed a mission in a real PV plant with a swarm of two UAVs.

4.1 Experimental setup

To first test the algorithms, we followed the simulation to real methodology that ensures safe real deployment of the system described in [20]. During this methodology we used only one UAV in order to have a minimal setup working. We started with a simulation development phase, followed by a precise Hardware-In-The-Loop (HITL) simulation and finishing with the industrial environment validation.

After testing the system from simulation to real scenario with a minimal setup, we added one UAV to the swarm, increasing the complexity of the mission. Real scenario is located at Repsol Technology Lab in Madrid.

For performing the real missions two different UAVs form the swarm were used to inspect the PV plants, these are the DJI M300, used in the single UAV setup, and DJI M350, used in the swarm setup together with the M300. Executing Aerostack2 software onboard, NVIDIA Xavier AGX boards have been mounted on top of each UAV.

For both real missions PV panel damage is simulated with an intentional hot-spot in the cells. Then, PV panel sends a request and the closest UAV executes the inspection. The results are sent to the panels (simulated and real) and the information is then managed by them. When no more requests are planned to be sent, the UAVs perform a return to land (RTL) operation.

Table 2 presents pairs time-distance from approach phase and inspection phase, main metrics used during the experiments.

Header	Description	Unit
t_{ap}	Time for approaching phase	Seconds
d_{ap}	Distance for approaching phase	Meters
t_{ins}	Time for inspection phase	Seconds
d_{ins}	Distance for inspection phase	Meters

Table 2: Time and distance metrics used during the experiments.

Moreover, through the ground station there are certain configurable parameters in the system, these are: speed, inspection altitude and safety altitude. Choosing this parameters depends on PV plant configuration and swarm capabilities. Table 3 details the parameters set during our experiments.

Drone	Speed (m/s)	IA (m)	SA (m)
M300	2.0	15	20
M350	2.0	15	30

Table 3: Setup for parameters in the experiments below. Speed: Maximun speed for flying. IA: Inspection Altitude, SA: Safety Altitude.

4.2 Simulation to Real Validation

For development purposes, we have first setup a simulation using Gazebo simulator. Simulation have been set imitating every possible parameter from real experiments; GPS initial positions, PV panel GPS positions, same configured speed and altitude and same communication protocol. After this, we shifted to DJI's HITL simulation. We decoupled

http://www.imavs.org/

http://www.imavs.org/

Gazebo's UAV simulation and used HITL instead. We then integrated the information from simulated panels in Gazebo together with the UAV HITL. Finally, we tested our system on a industrial environment with the same configuration used before.

Mission consists on using a single UAV (M300) that receives the requests from two panels multiple times. It receives a total of six requests. Panel 1 requests two times, then panel 2 make three requests, then at last, panel 1 sends one final request. The UAV then performs RTL operation. Path comparison between simulated and real mission can be seen in Figure 9. Time and distances measurements for both approach and inspection phases during the real mission can be found in Table 4.

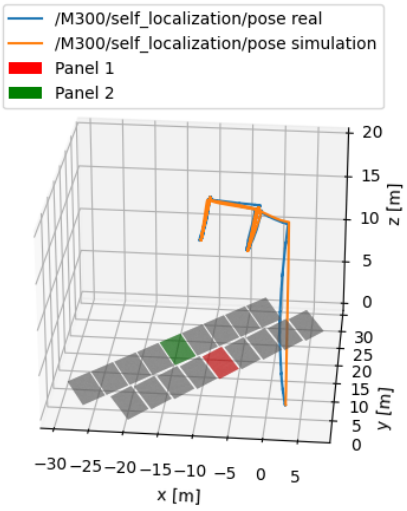


Figure 9: Path comparison between simulated path and real path. Mean square error between both paths is 0.19 meters.

Panel ID	$t_{ap}(s)$	$d_{ap}(m)$	$t_{ins}(s)$	$d_{ins}(m)$
Panel 1	30.23	29.34	15.08	10.35
Panel 1	0.41	0.11	14.27	10.87
Panel 2	5.97	10.07	21.85	10.89
Panel 2	0.53	0.09	17.12	11.22
Panel 2	0.55	0.10	15.18	11.06
Panel 1	6.05	10.36	18.15	10.99
Total	43.74	50.07	101.65	65.38

Table 4: Time and distance results from the industrial environment flights during simulation to real validation experiment. First row's approach phase includes takeoff distance and time.

Discussion

The simulation to real methodology ensured that no material or human resource was at risk. When looking at the trajectory comparison between the real and simulated mission, we see that we get a relatively low mean square error in meters ($MSE = 0.19$), increased only in the takeoff phase where the real UAV could not keep a straight path, but decreased in the rest of the mission where trajectory differences are within less than 0.1 meter error. Therefore, we conclude that simulation is reliable to test navigation and planning algorithms.

From time and distances tables, we can see that distance relates directly with time in the approach phase, however, as we can see in the first row where takeoff phase in included, the more vertical motion there is in a path, the longer it takes to reach. There is also high variability in inspection times due to variability introduced by inference times of the detection model.

4.3 PV plant swarm inspection

To validate the whole scope of our system, we increased the complexity from the previous experiments by adding an additional panel which is inspected by an additional UAV. By doing that, request management and mission planning complicate showing the strength of our system.

Mission consist on using a swarm of two UAVs (M300 and M350) and three panels making request wit no order. For this experiment, each panel requests only one inspection. Then, once inspections are finished, both UAVs perform RTL operation.

Figure 10 depicts paths followed by each UAV in this mission. Notice that the inspection altitude is the same for both UAVs, while the approach altitude differs to avoid collision if the trajectories cross each other during the approach phase. In this particular case, M350 is assigned to inspect panel 3, while M300 inspects panel 1 and 2 due to the proximity of each panel. Panel 1 is inspected first than panel 2 because its request arrived first to the ground station. Time and distances measurements for both approach and inspection phases can be found in Table. 5.

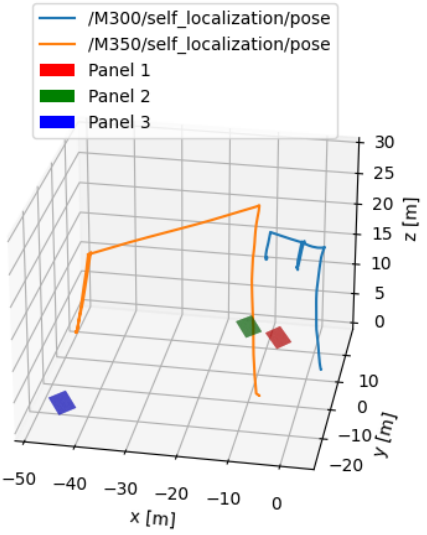


Figure 10: Swarm paths from real flights. M300 goes to panel 1 and panel 2. M350 goes to panel 3.

Panel ID	$t_{ap}(s)$	$d_{ap}(m)$	$t_{ins}(s)$	$d_{ins}(m)$
Panel 1	30.60	29.98	19.99	11.37
Panel 2	6.58	9.17	15.99	10.86
Panel 3	51.16	64.57	34.00	32.50

Table 5: Time and distance results from the swarm inspection. Panel 1 and 2 corresponds to M300. Panel 3 corresponds to M350. First and third row's approach phases include their respective UAV's takeoff distance and time.

Discussion

We can conclude that the IoT architecture integrates satisfactorily the intelligent PV plant monitoring subsystem with the aerial inspection subsystem, enabling a powerful dialog to cope with PV malfunctions. We also verified that the swarm oriented online planning is able to deal with multiple request arranging collision free inspection with a swarm of multiple UAVs. Finally, our onboard visual inspection and analysis was feasible to detect and report defects on real time. Inspection phases took on average 18 seconds, including a vertical descend (and ascend back) of 5 meters, a gimbal maneuver to center the panel on image, image capture and processing and reporting the result.

Comparing both UAVs inspections, we can confirm from the values in the last row of the results table that safety distance plays an important role in the inspection time, due to the vertical motion approach to inspect the panel. Difference between t_{ins} from panel 1 and 3 differs in $\sim$14 seconds, which is noticeable for an extra path of $\sim$11 meters only.

Our system offers a per-panel inspection approach. When a panel detects a failure and sends a request to the ground station, it takes less than 60 seconds for a UAV landed within 60 horizontal meters from the panel that made the request to be flying, positioned and ready for inspecting. Plus less than 35 seconds for it to inspect the panel, given a safety altitude of 30 meters and inspection altitude of 15 meters. For a current PV plant, if a panel fails, it won't be detected until inspection is scheduled and executed. Our approach ensures a more efficient PV panel energy production by addressing failures in an on-demand manner.

5 CONCLUSIONS

In this work, we introduced an autonomous UAV-based approach for on-demand inspections of photovoltaic panels. Unlike traditional manual path planning, which often relies on predefined paths or polygons, our system adapts in real time to environmental conditions and panel requests. Our method leverages online planning algorithms to optimize inspection routes dynamically, reducing overall inspection time and energy consumption while enhancing inspection accuracy.

Rather than inspecting entire arrays uniformly, our system focuses on individual panels, optimizing the resource allocation. When a panel signals a need for inspection (e.g., due to reduced output or anomalies), the UAV autonomously adjusts its flight path to address that specific panel. This targeted approach maximizes inspection accuracy while minimizing resource usage. Panels that perform within expected parameters receive fewer visits, while those with issues receive prompt attention. So that, this resource-efficient strategy enhances overall system reliability.

Integrating additional sensors (such as thermal cameras or multi-spectral imaging) further enhance our system. These sensors provide valuable data on panel temperature, cell health, and potential defects. By fusing this information with our online planning, we can achieve even more precise inspections.

We evaluated our system on a real photovoltaic facility, previously tested on simulation, performing successful per-panel inspection. The results showed our swarm oriented online planning algorithm working for multiple requests from the intelligent PV plant monitoring tool. Our onboard visual inspection and evaluation pipeline is able to detect faults on the panels on real time.

As part of the future work, a visual information method should be integrated in the inspection pipeline for better centering the panel center on image, rather than only using RTK and odometry measurements. Also, the proposed online planning algorithm should be improved to avoid cross paths between UAVs, instead of using different approach heights to avoid collisions. Finally, we aim to optimize the assign request-to-UAV algorithm to include priority information on the requests following certain criteria like severity of the possible fault or the distance to the panel.

ACKNOWLEDGEMENTS

This work has been supported by the project COPILOT ref. Y2020\EMT6368, funded by the Madrid Government under the R&D Synergic Projects Program. This work has also been supported by the project INSERTION ref. ID2021-127648OBC32, which was funded by the Spanish Ministry of Science and Innovation under the program "Projects for Knowledge Generating". In addition, this work has also been supported by the project RATEC ref: PDC2022-133643-C22 funded by the Spanish Ministry of Science and Innovation. This work has also been supported by the project GEMECOP ref. PID2022-139575OB-I00, funded by the Spanish Ministry of Science and Innovation. The work of the second author is supported under the Pre-Doctoral Program of Research and Technologic Innovation Department of Madrid Government. The work of the third author is supported by the Grant FPU20/07198 of the Spanish Ministry for Universities. We are very grateful to Repsol Technology Lab that has provided their facilities to run the real flight experiments.

REFERENCES

[1] Progress on competitiveness of clean energy technologies. Technical report, European Comission, 2023.

[2] Marco Colaprico, Maria Francesca de Ruvo, Giuseppe Leotta, Fabrizio Bizzarri, Silvano Vergura, and

http://www.imavs.org/

Francescomaria Marino. Dubio: a fully automatic drones & cloud based infrared monitoring system for large-scale pv plants. In *2018 IEEE International Conference on Environment and Electrical Engineering and 2018 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe)*, pages 1–5. IEEE, 2018.

[3] Gisele Alves dos Reis Benatto, Claire Mantel, Sergiu Spataru, Adrian Alejo Santamaria Lancia, Nicholas Riedel, Sune Thorsteinsson, Peter Behrensdorff Poulsen, Harsh Parikh, Søren Forchhammer, and Dezso Sera. Drone-based daylight electroluminescence imaging of pv modules. *IEEE Journal of Photovoltaics*, 10(3):872–877, 2020.

[4] Wolfgang Muehleisen, Gabriele C Eder, Yuliya Voronko, Markus Spielberger, Horst Sonnleitner, Karl Knoebl, Rita Ebner, Gusztav Ujvari, and Christina Hirschl. Outdoor detection and visualization of hailstorm damages of photovoltaic plants. *Renewable energy*, 118:138–145, 2018.

[5] Patrizio Manganiello, Marco Balato, and Massimo Vitelli. A survey on mismatching and aging of pv modules: The closed loop. *IEEE Transactions on Industrial Electronics*, 62(11):7276–7286, 2015.

[6] Future of solar photovoltaic. Technical report, International Renewable Energy Agency, 2019.

[7] Mahmoud Meribout, Varun Kumar Tiwari, Juan Pablo Pena Herrera, and Asma Najeeb Mahfoudh Awadh Baobaid. Solar panel inspection techniques and prospects. *Elsevier*, 2023.

[8] Aline Kirsten Vidal de Oliveira, Mohammadreza Aghaei, and Ricardo Rüther. Automatic inspection of photovoltaic power plants using aerial infrared thermography: A review. *Energies*, 15(6), 2022.

[9] A.M. Moradi Sizkouhi, S.M. Esmailifar, M. Aghaei, and M. Karimkhani. Robopv: An integrated software package for autonomous aerial monitoring of large scale pv plants. *Energy Conversion and Management*, 254:115217, 2022.

[10] Pia Addabbo, Antonio Angrisano, Mario Bernardi, Graziano Gagliarde, Alberto Mennella, Marco Nisi, and Silvia Ullo. Uav system for photovoltaic plant inspection. *IEEE Aerospace and Electronic Systems Magazine*, 33:58–67, 08 2018.

[11] Umesh Pruthviraj, Yashwant Kashyap, Effrosyni Baxevanaki, and Panagiotis Kosmopoulos. Solar photovoltaic hotspot inspection using unmanned aerial vehicle thermal images at a solar field in south india. *Remote Sensing*, 15(7), 2023.

[12] Luca Morando, Carmine Tommaso Recchiuto, Jacopo Calla, Paolo Scuteri, and Antonio Sgorbissa. Thermal and visual tracking of photovoltaic plants for autonomous uav inspection. *Drones*, 6(11), 2022.

[13] Zhipeng Xi, Zhuo Lou, Yan Sun, Xiaoxia Li, Qiang Yang, and Wenjun Yan. A vision-based inspection strategy for large-scale photovoltaic farms using an autonomous uav. In *2018 17th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, pages 200–203, 2018.

[14] Xiaoxia Li, Qiang Yang, Zhebo Chen, Xuejing Luo, and Wenjun Yan. Visible defects detection based on uav-based inspection in large-scale photovoltaic systems. *IET Renewable Power Generation*, 11(10):1234–1244, 2017.

[15] Ehab Salahat, Charles-Alexis Asselineau, Joe Coventry, and Robert Mahony. Waypoint planning for autonomous aerial inspection of large-scale solar farms. In *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, volume 1, pages 763–769, 2019.

[16] Yifan Ding, Rui Cao, Siming Liang, Fengyang Qi, Qiang Yang, and Wenjun Yan. Density-based optimal uav path planning for photovoltaic farm inspection in complex topography. In *2020 Chinese Control And Decision Conference (CCDC)*, pages 3931–3936, 2020.

[17] Xuejing Luo, Xiaoxia Li, Qiang Yang, Fengjie Wu, Duo Zhang, Wenjun Yan, and Zhipeng Xi. Optimal path planning for uav based inspection system of large-scale photovoltaic farm. In *2017 Chinese Automation Congress (CAC)*, pages 4495–4500, 2017.

[18] Miguel Fernandez-Cortizas, Martin Molina, Pedro Arias-Perez, Rafael Perez-Segui, David Perez-Saura, and Pascual Campoy. Aerostack2: A software framework for developing multi-robot aerial systems. *arXiv preprint arXiv:2303.18237*, 2023.

[19] Javier Rodriguez-Vazquez, Inés Prieto-Centeno, Miguel Fernandez-Cortizas, David Perez-Saura, Martin Molina, and Pascual Campoy. Real-time object detection for autonomous solar farm inspection via uavs. *Sensors*, 24(3), 2024.

[20] Rafael Perez-Segui, Pedro Arias-Perez, Javier Melero-Deza, Miguel Fernandez-Cortizas, David Perez-Saura, and Pascual Campoy. Bridging the gap between simulation and real autonomous uav flights in industrial applications. *Aerospace*, 10(9), 2023.

http://www.imavs.org/

Flying a Quadrotor with Unknown Actuators and Sensor Configuration

T.M. Blaha*, E.J.J. Smeur, B.D.W. Remes and C.C. de Visser
Delft University of Technology, Kluyverweg 1, 2629 HS Delft, The Netherlands

http://www.imavs.org/

ABSTRACT

Though control algorithms for multirotor Unmanned Air Vehicle (UAV) are well understood, the configuration, parameter estimation, and tuning of flight control algorithms takes quite some time and resources. In previous work, we have shown that it is possible to identify the control effectiveness and motor dynamics of a multirotor fast enough for it to recover to a stable hover after being thrown 4 meters in the air. In this paper, we extend this to include estimation of the position of the Inertial Measurement Unit (IMU) relative to the Center of Gravity (CoG), estimation of the IMU rotation, the thrust direction of all motors and the optimal combined thrust direction. In order to guarantee a correct IMU position estimation, two prior throw-and-catches of the vehicle with spin around different axes are required. For these throws, a height as low as 1 meter is sufficient. Quadrotor flight experimentation confirms the efficacy of the approach, and a simulation shows its applicability to fully-actuated crafts with multiple possible hover orientations.

1 INTRODUCTION

Drones are used for an ever increasing number of applications, with variations in the number and placement of propellers, the craft's inertia and sensor location and orientation. Careful controller design is needed to ensure new platforms are stable and demonstrate good flight performance. However, the process of developing, testing, and fine-tuning flight control algorithms for new drones is both time-consuming and expensive, which poses a significant barrier to the advancement of new drone applications.

In recent work [1], we have shown that it is possible to learn all required control parameters in the span of a single throw, such that a multirotor learns to fly without prior knowledge of its motor dynamics time constant, actuator position and effectiveness. In that paper, two important assumptions were made. First it was assumed that the motors were providing upward thrust in a body frame, simplifying the position control to that of a standard multirotor. The motors may not always be mounted in this configuration, or in the case of fully or overactuated multirotors, independent thrust can be generated in any direction, allowing multiple hover attitudes [2]. The second assumption was that the Inertial Measurement Unit (IMU) orientation and position offset relative to the Center of Gravity (CoG) were known. This assumption was needed, because of the high angular rates the vehicle obtains during throwing and excitation, which necessitate compensation of the acceleration measured by the IMU to transform it to the CoG.

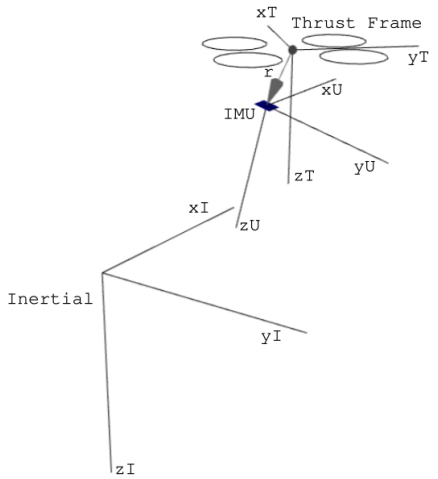


Figure 1: Coordinate Systems: Identifying the transformation between thrust frame T and IMU frame U from less than 1 second of flight data is the main contribution of this work.

Online estimation of the position of the CoG relative to the IMU was addressed in [3], but this was done relying on actuator inputs with a predetermined effectiveness. Svacha et al. [4] implemented a 32 state Unscented Kalman Filter (UKF) estimating all mass properties, including IMU offset. However, this was done with offline data, with known mass and motor locations. Lukacs and Lantos [5] provide an algorithm for determining the position of the CoG together with a force model for a sailplane, but practical results are not provided. [6] estimated the moments of inertia, CoG position, and aerodynamic parameters of a quadrotor model using a Minimum Likelihood Estimator, but this can only be

*Email address(es): {t.m.blaha; e.j.j.smeur; b.d.w.remes; c.c.devisser}@tudelft.nl
Data: 10.4121/0530be90-cc6c-4029-9774-670657882906
Code: github.com/tudelft/indiflightSupport/tree/iros_imav_2024

http://www.imavs.org/

run offline. The typical approach in the literature seems to be to estimate the CoG offset at the same time as the control effectiveness, which either requires some assumptions on the geometry of the drone, or extends the estimation time significantly.

In this paper, we extend the work of [1] to unknown IMU orientation and position, and unknown thrust directions of the actuators. However, the basic assumption that the vehicle has to be capable of static hover is maintained. The only user input to the system is two throw-and-catches of about 1 meter height with sufficient spin and one throw-and-fly of around 3.5 meter height. The applicability of the algorithm is shown through real life experiments with a quadrotor.

2 FRAMES OF REFERENCE

We distinguish 3 coordinate frames, illustrated in Figure 1, and relate them using unit quaternions. The frame of the IMU with respect the inertial frame is described by q_I^U. The "thrust frame" T is defined with its origin in the center of gravity of the vehicle, and the z^T-axis pointing opposite a chosen thrust axis. This axis is obvious for quadrotors, but e.g. for fully actuated hexarotors that can hover in many different orientations[2], the choice may not be obvious, so that we define $-z^T$ as the axis of *most efficient* thrust generation capacity while simultaneously not causing any body-moments. The choice of x^T is commonly taken as the front of the vehicle, but in our study this choice is arbitrary.

The thrust frame relates to the IMU frame with q_U^T, and to the inertial frame with the intrinsic rotation sequence $q_I^T = q_I^U q_U^T$.

A problem with the arbitrary definition of the x^T and y^T axes, is that the yaw angle needs to be known in order to control the position and velocity of a multirotor. We see two solutions for this problem.

The first is an Extended Kalman Filter (EKF) that estimates the heading by fusing Global Navigation (GNSS) measurements and IMU acceleration measurements projected onto the inertial xy-plane. The second is to include a magnetometer, of which the axes are aligned with the IMU.

In this study we only consider IMU rotations around an axis in the thrust-xy plane. This guarantees that a desired inertial accelerations can be translated to the correct attitude changes for closed loop control.

3 METHODOLOGY

This section gives a brief overview of the state estimation, controller structure and system identification onboard of the UAV. Then our method to determine the IMU location and the thrust frame is explained with focus on real-time implementation.

3.1 Controller Summary and Effectiveness identification

The same controller and estimator structure of [1] is also used in this work. A short summary:

A complementary attitude filter is used to estimate q_I^U. The yaw angle computed by an external motion capture system is fused into this estimate.

An Incremental Nonlinear Dynamic Inversion (INDI) controller is used to give motor commands to achieve a desired q_I^T, which in turn is computed by a Nonlinear Dynamic Inversion (NDI) position controller.

The identification of the effectiveness matrix and motor model for the INDI controller results from a Recursive Least Squares (RLS) implementation based on the measured angular rate derivatives and specific forces in response to a sequential open-loop excitation of each motor.

3.2 IMU location estimation

The inertial acceleration a_r of a point r expressed in the coordinates of a local, rotating frame T is given by:

$$a_r = a_T + \ddot{r} + 2\Omega_T \times \dot{r} + \dot{\Omega}_T \times r + \Omega_T \times (\Omega_T \times r) \,, \quad (1)$$

where a_T is the inertial acceleration of the frame, $r,\ \dot{r},\ \ddot{r}$ is the location, velocity and acceleration of r relative to frame T, and $\Omega_T,\ \dot{\Omega}_T$ the rotation rate and acceleration of frame T. All quantities are expressed in the coordinates of frame T.

A rigidly mounted IMU is now placed at r, and the origin of the rotating frame is chosen at the vehicle's center of gravity: $\dot{r} = \ddot{r} = 0$. Since accelerometers do not measure gravity (but rather all other specific forces), its reading for a free-tumbling rigid body on which no external forces act will follow this reduced equation with $a_T = 0$:

$$a_r = \dot{\Omega}_T \times r + \Omega_T \times (\Omega_T \times r) \quad (2)$$

$$\underbrace{a_r}_{y_i} = \underbrace{\begin{pmatrix} -\Omega_y^2 - \Omega_z^2 & \Omega_x\Omega_y - \dot{\Omega}_z & \Omega_x\Omega_z + \dot{\Omega}_y \\ \Omega_x\Omega_y + \dot{\Omega}_z & -\Omega_x^2 - \Omega_z^2 & \Omega_y\Omega_z - \dot{\Omega}_x \\ \Omega_x\Omega_z - \dot{\Omega}_y & \Omega_y\Omega_z + \dot{\Omega}_x & -\Omega_x^2 - \Omega_y^2 \end{pmatrix}}_{X_i} \underbrace{\begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix}}_{r} . \quad (3)$$

This equation is linear in the parameters r, and so if $\Omega,\ \dot{\Omega},\ a_r$ are measured during such a tree-tumble, the observations X_i and y_i can be aggregated in matrix X and vector y. Least Squares (LS) can then be used to obtain an estimate of r.

The conditions of a tumbling rigid body with minimal external forces (e.g. from aerodynamic drag) can be achieved by throwing it in the air with sufficient angular velocity. One single throw of the vehicle would be ideal, however it is generally not enough to reliably fit r: if the vehicle is rotating around a single axis (without loss of generality, assume the axis $(0,\ 0,\ 1)$), then the resulting equation is

$$a_r = \begin{pmatrix} \Omega_z^2 & -\dot{\Omega}_z & +\dot{\Omega}_y \\ \dot{\Omega}_z & -\Omega_z^2 & -\dot{\Omega}_x \\ -\dot{\Omega}_y & \dot{\Omega}_x & 0 \end{pmatrix} \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} \,, \quad (4)$$

http://www.imavs.org/

and so if $(0,\ 0,\ 1)$ happens to be close to a principle inertial axis, then $\dot{\Omega}$ is expected to be small and r_z cannot be estimated. It is later experimentally illustrated that two throws about significantly different axes provide enough excitation.

3.3 Thrust Direction Estimation

Instead of estimating the IMU's orientation with respect to some other body-fixed frame, we estimate the direction in which force can be generated with the motors, without producing body-moments. For conventional quadrotors there is only one such direction (coinciding with the axes of each motor), but many more directions are possible for configurations with more actuators and tilted rotors. If we knew a suitable such direction (e.g. the most power-efficient one), we can use this to construct a reference frame in which a conventional multirotor controller can operate.

Recall that we assume the multirotor configuration to be able to achieve steady-state hover. We now approximate the specific forces and torques acting on the vehicle as linear combinations of the motor input $u \in \mathbb{R}^m$, where m is the number of motors on the vehicle:

$$\begin{pmatrix} f \\ \tau \end{pmatrix} = \begin{pmatrix} G_{1,f} \\ G_{1,\tau} \end{pmatrix} u \,. \tag{5}$$

Here, the $f \in \mathbb{R}^3$ and $\tau \in \mathbb{R}^3$ are the specific forces and torques, respectively, and $G_{1,f} \in \mathbb{R}^{3\times m}$ and $G_{1,\tau} \in \mathbb{R}^{3\times m}$ are the corresponding steady-state effectiveness matrices. In [1], these matrices are identified locally using the increments of u and $(f,\ \tau)^T$, but here we assume that they hold in the global sense of Equations 5. Note that Equation 5 is only valid in steady-state when $\dot{u} = 0$.

The most efficient distribution of control effort in hover can be defined as the one that minimizes some square norm of u, while ensuring (1) there are no torques acting on the vehicle ($\tau = 0$), (2) the gravity is canceled by the forces ($||f|| = g$), and (3) the bounds on u are respected. Introducing the diagonal actuator weighing matrix W, these statements are summarized in the non-convex quadratically constrained quadratic problem

$$\begin{aligned} \min_{u} \quad & u^T W u \\ s.t. \quad & u^T (G_{1,f}^T G_{1,f}) u = g^2 \\ & G_{1,\tau} u = 0 \\ & \underline{u} \le u \le \overline{u} \,. \end{aligned} \tag{6}$$

If a solution $u = u^*$ exists, then the vehicle is indeed capable of static hover and the most efficient direction to generate thrust without causing rotations is $d = G_{1,f} u^*$.

General quadratic programming solvers readily solve this problem, however, their implementation in embedded hardware is difficult, and computation time is slow and convergence time may not be guaranteed. If G_1 changes, or at least its estimate, then Problem 6 would have to be solved again and again, which quickly becomes infeasible.

Real-Time Implementation We now introduce a heuristic and a simplification that allows development of an efficient algorithm to solve Problem 6 in sufficient cases.

Firstly, assume that the bounds $\underline{u} \le u \le \overline{u}$ are always fulfilled if the first two constraints are met, and can thus be dropped. This may cause the found solution to be infeasible if some motors are at their limit or not used at all at true solution u^*. The rationale for this simplification is that it is unlikely that a vehicle capable of static hover requires motors at maximum or minimum thrust even in its most efficient hovering attitude.

Secondly, assuming identical motors, impose $W = I$. This will cause sub-optimal hover distributions when the motors do not have the same power consumption for a given input u.

The algorithm follows these steps

1. Introduce new variables $\eta \in \mathbb{R}^{m-3}$ by calculating an orthogonal basis N_τ for the $(m-3)$-dimensional nullspace of $G_{1,\tau}$ (full-rank is guaranteed by the assumption that the vehicle can achieve static hover). This can be efficiently achieved by a column-pivoting complete QR decomposition of $G_{1,\tau}^T$, e.g. with the LAPACK routine SGEQP3[1].

2. Now, any choice of $u = N_\tau \eta$ satisfies the constraint $G_{1,\tau} u = 0$, and what remains is

$$\begin{aligned} \min_{\eta} \quad & \eta^T N_\tau^T W N_\tau \eta \\ s.t. \quad & \eta^T (N_\tau^T G_{1,f}^T G_{1,f} N_\tau) \eta = g^2 \,. \end{aligned} \tag{7}$$

   Defining $H \triangleq N_\tau^T W N_\tau$, and $A \triangleq N_\tau^T G_{1,f}^T G_{1,f} N_\tau$, and introducing the Lagrange multiplier λ, we can write the KKT system of Equation 7 as

$$\begin{aligned} 2H\eta + 2\lambda \cdot A\eta &= 0 \\ \eta^T A \eta &= g^2 \end{aligned} \tag{8}$$

3. It is now clear why we imposed $W = I$: N_τ is an orthogonal basis for the nullspace, due to our choice of the QR factorization. Consequently, $H = I$ as well, and the first line of Equation 8 is an eigen-problem $Av_i = \tilde{\lambda}_i v_i$, where we explicitly, but without loss of generality assume $||v_i|| = 1$. Now we know that $\eta^* = k v_i$ with one of the eigenvectors of A and some scaling factor k, which can be computed by substituting this relation into the second line of 8:

$$k^2 v_i^T \underbrace{A v_i}_{\tilde{\lambda}_i v_i} = g^2 \implies k = \frac{g}{\sqrt{\tilde{\lambda}_i v_i^T v_i}} = \frac{g}{\sqrt{\tilde{\lambda}_i}} \,. \tag{9}$$

[1] netlib.org/lapack/explore-html/d0/dea/group__geqp3



Figure 2: Compact 3-inch quadrotor used for the flight experiments.

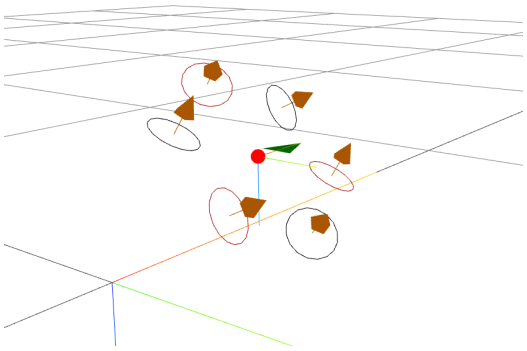


Figure 3: Fully-actuated hexarotor that can hover in multiple orientations.

Consequently, $u^* = \pm N_\tau \eta^* = \pm N_\tau k v_i = \pm N_\tau g/\sqrt{\tilde{\lambda}_i} v_i$, where the sign must be chosen so that the components of u^* are within $\underline{u} \leq u \leq \overline{u}$. Following our heuristic, if this is impossible, then the vehicle is very likely not capable of static hover in the first place. This relation also shows that we require not just any, but the largest eigen-pair (or one of the largest eigen-pairs, if there are multiple equal ones) of A, as that results in the lowest magnitude of u^*.

4. Since $A \succ 0$, we can compute the largest eigen-pair of A using the power iteration $v_{j+1} = \frac{Av_j}{||Av_j||}$. In practice an iteration limit should be imposed, and since the parameters of the problem are not expected to change rapidly, subsequent invocations of step 4. will continue to improve the guess.

5. Finally, the axis of most efficient thrust generation is then $d = G_{1,f} u^*$, where $u^* = \pm N_\tau g/\sqrt{\tilde{\lambda}} v_i$, with $\pm$ and v_i as determined above. Then, q_U^T can finally be assembled as the shortest rotation that maps d to $(0,\ 0,\ -g)$.

Most of the computation time of the algorithm above is expected in the computation of the QR-decomposition which increases linearly with the amount of motors m^2. The subsequent matrix calculations increase with $(m-3)^2$.

3.4 *Experiment Design – Quadrotor*

The estimation and control algorithm is verified experimentally, in the same fashion as done in [1], by implementing the computations in the firmware INDIflight[2] onboard of an STM32H743 flight controller (480 MHz clockspeed), and using gyroscope and accelerometer data from a TDK InvenSense ICM-42688-P IMU. The vehicle is pictured in Figure 2. The update rate of the attitude control and actuator effectiveness estimation is $2kHz$. Even though it is not necessary to run the thrust frame estimation at the same rate, this is still performed for ease of comparison of the runtimes later. For safety reasons, the quadrotor is launched and given an initial rotation of $\Omega_0 = (400,\ 400,\ 100)^T$ deg/s to simulate a human throw. The invariance of the estimations to this initial condition has been shown in [1].

In order to test different IMU orientations without making changes to the hardware, the IMU measurements are rotated by the inverse of $q_U^T = 0.837 - 0.491i + 0.242j$, which corresponds to Euler angles of $-15°$ Yaw, $24°$ Pitch, $-64°$. As explained in the limitation of Section 2, this rotation results in correct yaw alignment when rotated back using the algorithm of Section 3.3.

Though it is entirely possible to do the estimation of the IMU location r onboard, for the demonstration of this work it has been performed offline with logged sensor measurements, and was then manually programmed into the memory of the flight controller. Onboard estimation of the parameter would merely require detection of the 2 required throws and casting the Least Squares (LS) estimator as a Recursive Least Squares (RLS) estimator to eliminate the memory requirements of the offline LS.

3.5 *Simulation Design – Fully Actuated Hexarotor*

Steps 3 through 5 of the algorithm in Section 3.3 only need to involve the matrix algebra if the rotational nullspace N_τ spans multiple dimensions. In the case of a quadrotor, $\dim(N_\tau) = 1$ and steps 3-5 are trivial.

The hexarotor ($\dim(N_\tau) = 3$) shown in Figure 3 has an arrangement such that there is one energy-optimal hover orientation, obvious by inspection, but many other suboptimal ones [2]. To test our algorithm, a simulation model has been created with the added difficulty that the entire actuator constellation is rotated by $45°$ with respect to the IMU (the green arrow in Figure 3). This illustrates again the equivalence of unknown IMU orientation, and unknown actuator locations; our approach does not need an explicitly defined "body-frame".

The simulation is run with the same control code as in the

[2]github.com/tudelft/indiflight

http://www.imavs.org/

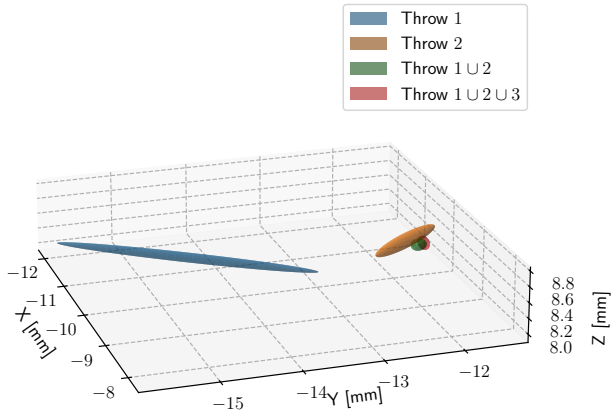


Figure 4: 95% percent confidence ellipsoids of the IMU offset parameter for a sequence of throws, where $1 \cup 2$ stands for the union of the data obtained from throws 1 and 2. Least Squares fails for just 1 throw, but succeeds for multiple.

flight-experiment, and may be reproduced using Docker[3].

4 RESULTS

All data used in this work is available for reproduction in a dataset[4].

4.1 IMU location

Figure 4 shows the 95% percentile confidence ellipsoids for the parameters $\hat{r}$ after LS fitting, defined by $(r - \hat{r})^T \Sigma^{-1} (r - \hat{r}) < iCDF(\chi^2_3, 0.95)$. Here, $\Sigma \approx \text{SE}^2 \cdot (X^T X)^{-1}$ and the standard error of the residuals $\text{SE}^2 = ||y - X\hat{r}||_2^2/(N-3)$ [7].

These ellipsoids capture the effects of correlation in the problem data X and the residual error variance in the predictions $y - X\hat{r}$. Systematic errors like those arising from, e.g. inaccurate scaling of the IMU measurements, are not captured.

Using only the observations from a single throw yields uncertain estimates in one direction, as indicated in Section 3.2. However, combining two throws with spin around two different axes reduces the confidence bounds to the sub-millimeter range and is not much improved by adding data from a third throw. The remaining uncertainty may be due to sensor bias, aerodynamic drag being a factor ($a^T \neq 0$), and the individual test masses of the 3 Micro Electro-Mechanical System (MEMS) accelerometers not being exactly in the same location.

4.2 Flight Experiment Results – Quadrotor

The recovery of the vehicle and attaining a position setpoint follows a similar trajectory to the results shown in [1]. The excitation and system response are shown on the left in Figure 5. Due to the alignment of the IMU, each motor gives rise to specific forces in all three axes. The right side of Figure 5 shows the identification of the translational part of the effectiveness matrix $G_{1,f}$, where every vertical dashed line corresponds to the start of the excitation of the next actuator. Once all actuators have been triggered (roughly 300ms into the sequence), the computation of the algorithm determining the thrust direction (Section 3.3) runs to completion, as $G_{1,\tau}$ now has a nullspace of dimension greater than 0.

The estimated and true alignment of the IMU is given in Table 1. The alignment is sufficiently accurate to not affect manual or automatic position control performance.

	Ground truth	Estimated
Roll	-15°	-14.1°
Pitch	24°	23.7°
Yaw	-64°	-61.0°

Table 1: Quadrotor Flight Experiment: Estimated rotation of the IMU compared to the ground truth.

4.3 Execution Timings

All execution on the Flight Controller is single-thread, which means that execution time can be measured accurately, with only interrupt handling being a source of error. Figure 7 shows the cumulative execution times of the components of the system identification and thrust axis determination algorithms. The compilation was done with `gcc 10.3.1` at optimization level `-O3`.

The thrust axis estimation adds only a small delay, amounting to an additional $2kHz \cdot 20e-6s = 4\%$ CPU load, over the other identification components present in [1].

4.4 Simulation Results – Hexarotor

Also in the simulated throw-to-hover, the results show that the optimal thrust frame is estimated correctly at the end of the fitting process. In the ground truth, this frame of most efficient hover has been rotated in roll by 45°, and so the attitude quaternion that our algorithm seeks is $q_U^T = 0.924 + 0.383i$. The lower right subplot in Figure 6 shows that this is reached at the time the last actuator has been excited. In particular, our method found $\hat{q}_U^T = 0.930 + 0.367i - 0.026j$, which is shown in Euler angles in Table 2.

	Ground truth	Estimated
Roll	45°	43.1°
Pitch	0°	-2.74°
Yaw	0°	-1.08°

Table 2: Hexacopter Simulation: Estimated rotation of the IMU compared to the ground truth.

[3] github.com/tudelft/indiflightSupport/tree/iros_imav_2024
[4] 10.4121/0530be90-cc6c-4029-9774-670657882906

http://www.imavs.org/

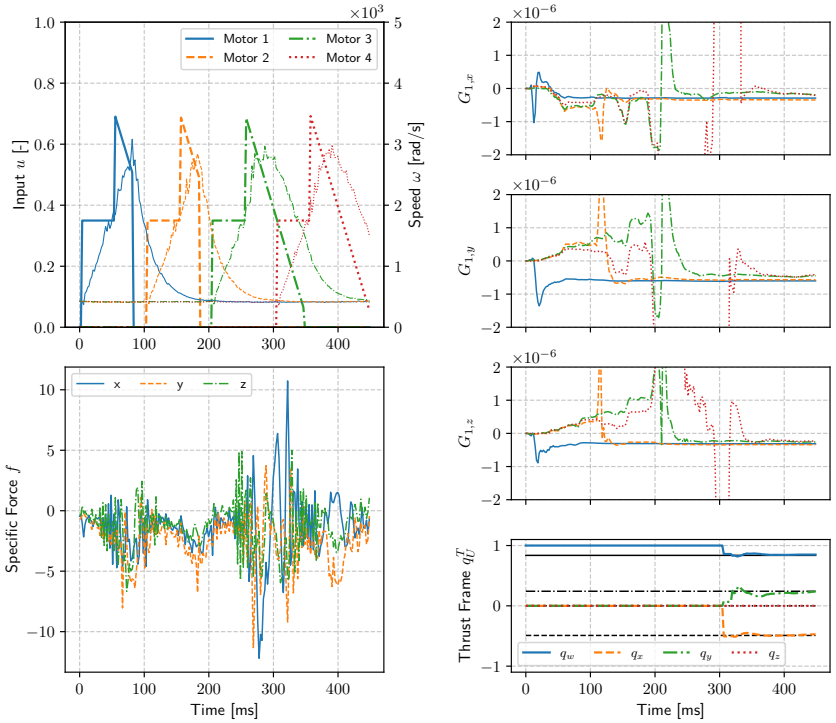


Figure 5: Flight experiment: Inputs u, motor response ω, accelerometer measurements f and fitting of the linear part of the Effectiveness Matrix ($G_{1,xyz}$). The last plot shows that the true IMU alignment is approached closely, once the last motor is excited.

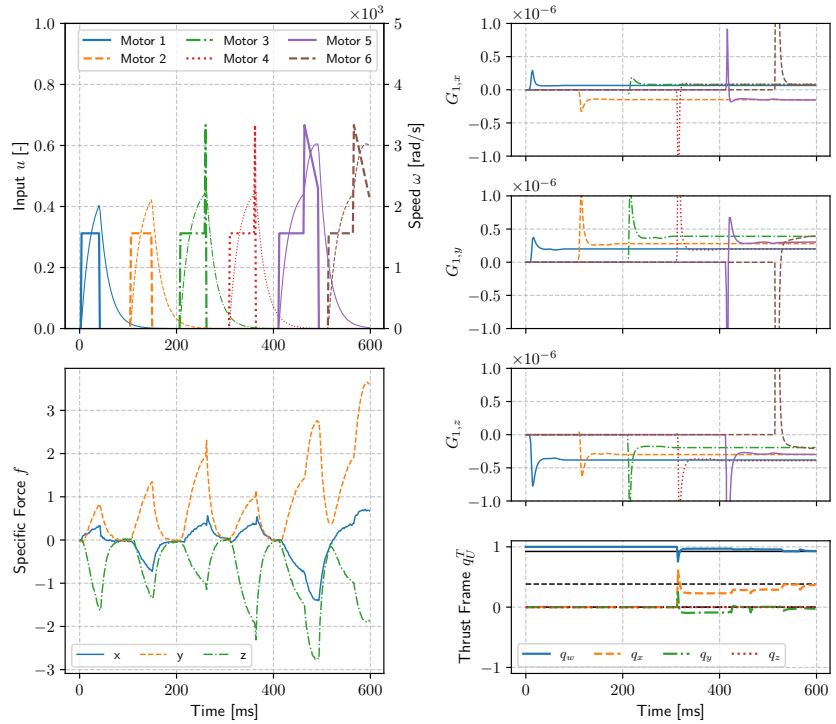


Figure 6: Hexarotor simulation: The parameters of the 6 actuators are identified in the span of 600ms. The specific-force part of G_1 (xyz-components shown on the top right for all 6 actuators) show the significant contributions in all IMU axes, not just z. From these measurements, our method estimates the optimal thrust frame.

http://www.imavs.org/

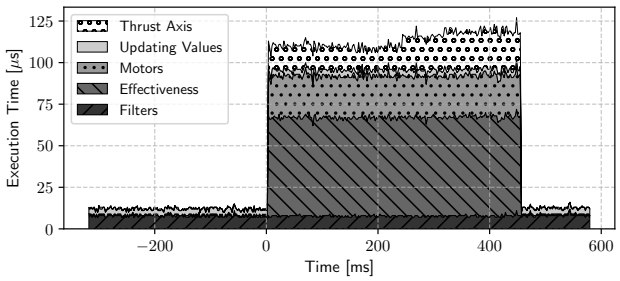


Figure 7: Cumulative execution time on the flight controller show that 2kHz update rates are feasible for quadrotors.

5 CONCLUSION

In this study, we successfully demonstrated attitude control of a quadrotor where the motors' location and strength are unknown, as well as the alignment and location of the IMU. The location of the IMU has been estimated from two prior throw-and-catches, after which the quadrotor was able to recover from being launched on a 3.5m parabolic trajectory.

Recursive Least Squares was used to identify the motors' rotational and linear effectiveness in-flight, which in turn were used to calculate an optimal thrust direction in the frame of the IMU which can be actuated without giving rise to rotations. The latter is the main contribution of this paper and enables position control if an external source of position and heading (GPS+magnetometer, optical motion capture) is available and the alignment of the IMU q_U^T is restricted to yaw-free rotations. The presented algorithm computes in 20μs on consumer-grade flight control hardware, however computation time will grow with $(m-3)^2$, where m is the number of actuators.

The ability of our method to compute the most efficient hover direction, as opposed to just any, was confirmed in a simulation of a fully actuated hexarotor.

In future work, position control should be enabled even in the cases where the IMU-to-hover frame alignment includes yaw components, either by including a magnetometer sensor aligned with the IMU, or by calculating heading from comparing the vehicles GPS-track with the acceleration measured onboard.

Furthermore, the dependence on prior throws may be eliminated by eliminating the need to know the IMU location before identification of the motors' effectiveness. This could be achieved by providing additional excitation in the nullspace of $G_{1,\tau}$ once rotations have been arrested. In this case, the accelerometer will measure the true specific forces, as every point on the vehicle experiences the same acceleration.

ACKNOWLEDGEMENTS

The authors would like to thank Elijah Hao Wei Ang for his insights.

REFERENCES

[1] Till Blaha, Ewoud Smeur, and Bart Remes. Control of Unknown Quadrotors from a Single Throw. *arXiv*, arXiv:2406.11723, 2024.

[2] Sujit Rajappa, Markus Ryll, Heinrich H. Bülthoff, and Antonio Franchi. Modeling, control and design optimization for a fully-actuated hexarotor aerial vehicle with tilted propellers. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4006–4013, May 2015.

[3] Emil Fresk, David Wuthier, and George Nikolakopoulos. Generalized center of gravity compensation for multirotors with application to aerial manipulation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4424–4429, September 2017.

[4] James Svacha, James Paulos, Giuseppe Loianno, and Vijay Kumar. IMU-Based Inertia Estimation for a Quadrotor Using Newton-Euler Dynamics. *IEEE Robotics and Automation Letters*, 5(3):3861–3867, July 2020.

[5] L. Lukacs and B. Lantos. Nonlinear dynamic model identification of airplanes including inertial parameters and force-Torque relations based on real flight data. In *SACI 2015 - 10th Jubilee IEEE International Symposium on Applied Computational Intelligence and Informatics, Proceedings*, pages 393–398, 2015.

[6] M. Burri, J. Nikolic, H. Oleynikova, M.W. Achtelik, and R. Siegwart. Maximum likelihood parameter identification for MAVs. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2016-June, pages 4297–4303, 2016.

[7] Larry Wasserman. Lecture 18: Tests and Confidence Sets for Multiple Coefficients. Lecture Series, Carnegie Mellon University, Pittsburgh, PA 15213, USA, 2017.

http://www.imavs.org/

Novel Flapping Wing Micro Aerial Vehicle Design Featuring a Bio-inspired Abdomen Mechanism

Jiaxun Leng*, Yuk Wa Steven Ng*, Shun Tang*
info.fwmav@gmail.com
Imperial College London, South Kensington, London

[1]

http://www.imavs.org/

ABSTRACT

This paper introduces a novel flapping wing micro aerial vehicle (FWMAV) design that integrates a bio-inspired abdomen mechanism to enhance flight stability and maneuverability. By mimicking the abdominal mechanism observed in insects like hawkmoths, our design aims to leverage the abdomen's ability to act as an aerodynamic rudder, shifting the center of mass and redirecting lift forces to achieve instant control responses. The proposed model combines an upper X-wing configuration with a servo-actuated abdomen housing the majority of the vehicle's electronics. This paper will present the equation of motions derived from the Euler-Lagrange Equation and the abdomen's stability properties, followed by experimentally measured data from a computer vision velocity tracking setup and a simulation model to act as a proof of concept and explore the response of the model realistically. This novel design has the potential to significantly improve the performance of FWMAVs in dynamic and unpredictable environments.

1 INTRODUCTION

Bio-inspired flapping wing micro air vehicle (FWMAV) has slowly gained research interest in the MAV community and a few general configurations of FWMAV prototypes have emerged such as the two-wing hummingbird-like configuration[1], X-Wing configuration[2] and tailless configuration[3][4]. However, there still exists a major trade-off between general stability and maneuverability. Recent studies of passive stability and efficient control mechanisms for insect flight might have shined some new light on the issue. The inherently unstable nature of flapping-wing flight can be actively controlled through rapid reactions or passively stabilised with high body drag[5]. Computational studies highlight the complexity of nonlinear interactions affecting stability[6]. The integration of aerodynamics, sensing, and control in insect-scale flapping-wing flight is crucial for achieving stability[7]. Research on the longitudinal dynamic flight stability of insects such as hoverflies and craneflies emphasise the critical role of parts other than the control surfaces[8], particularly the abdomen, in passive stabilisation[9], especially in gusty conditions[10], and hidden stabilisation mechanisms like vibrational control are essential for this process[11]. Traditional approaches often rely on wing stroke plane modulation or the integration of fixed control surfaces to achieve the desired stability. For instance, Nguyen and Chan (2018) developed a tailless FWMAV that utilised wing stroke plane modulation to control flight dynamics, demonstrating effectiveness but at the cost of increased drag each time the status of the wing changes[4]. Similarly, Karasek et al. (2018) explored the use of torque coupling in rapid banked turns, relying heavily on fixed aerodynamic surfaces, which also introduced additional drag and reduced overall efficiency each time the control is activated[3]. De Wagter et al. (2014) emphasized that conventional control surfaces, though functional, can be a limiting factor in achieving optimal flight efficiency in small-scale aerial vehicles[2].

The abdomen of some insects, however, can be one of the pivotal roles not only in maintaining flight stability, especially under gusty conditions but also in maneuverability by acting as an aerodynamic rudder that shifts the centre of mass and redirects lift forces[9]. Recent studies have shown that the hawkmoth (Manduca sexta) uses its abdomen to achieve pitch stability through fine-tuned adjustments that are synchronized with wing beats[10]. The abdominal control mechanisms in moths are crucial for maintaining stable hovering flight, and operating near the edge of stability for quick transitions[10]. Incorporating these insights into the design of flapping wing MAVs can significantly enhance their stability and gust resistance.

This research proposes and demonstrates the

[1] All authors contributed to this work equally and are ranked based on alphabetical order

effectiveness of the novel abdomen control both in flight stability as well as manoeuvrability achieved through centre of mass shifting but also opens new avenues to explore for bio-inspired control algorithms, such as the vibrational control particularly leveraging the active role of the abdomen in enhancing stability. The paper is broken down into three main parts: the first part introduces the mechanical design and actuation of the abdomen; second part introduces the equation of motions of the FWMAV. Finally, the experimental and simulation setup is introduced, where explains our planned experiments and data acquisition.

2 THE DESIGN CONCEPT

The prototype features an upper body of an X-wing configuration with an abdomen lower body, which hosts the electronics that accounts for 65% of the bodyweight. The wing and flapping mechanism follows the same principles as many established works [2][12] and the aerodynamic performance is well documented [13].The FWMAV weighs 40 grams with a wingspan of 33 cm and a body length of 30 cm, dimensions carefully chosen to balance aerodynamic performance and structural integrity. The wing flapping mechanism is capable of achieving a flap angle of 105 degrees, enabling significant lift and thrust generation. The loaded frequency was optimised to 13 Hz to accommodate the aerodynamic forces experienced during flight.

The abdomen part of the model can have potential advantages over the conventional flapping wing drones by dynamically adjusting the position of centre of gravity (CG), providing an additional degree of freedom in the control design. The servo control would also allow for the ability to fine-tune the CG for better control. This is particularly beneficial for flapping wing drones, where quick manoeuvres are often necessary to maintain stability and avoid obstacles such as in urban areas or forests where there is higher density of obstacles. Traditional fixed-wing aircraft can introduce drag. The adjustable abdomen removes the dependency on these surfaces, potentially leading to more efficient flight dynamics with less drag.

2.1 The Abdomen Mechanism

Inspired by the anatomical structure of an insect's abdomen, a mechanism was designed to translate the rotary motion of two servo motors into the rotation of the MAV's abdomen about the pitch and roll axes, which are offset by 12.5 mm. This design effectively decouples the two axes, ensuring that motion in one axis does not interfere with the other. The servos, which are the heaviest individual components of the MAV, are strategically positioned at the bottommost part of the abdomen. This placement, along with any additional electronics, serves to lower the overall center of gravity.

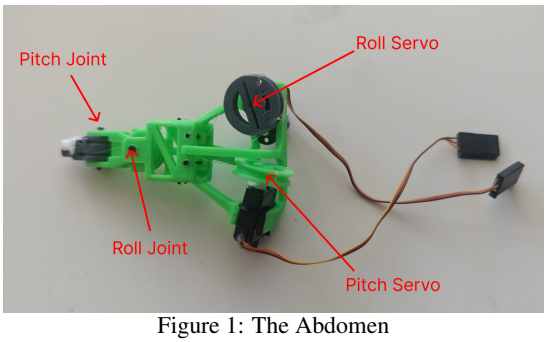


Figure 1: The Abdomen

Lowering the center of gravity confers several advantages to the flapping-wing drone. It enhances the MAV's overall stability, particularly by reducing its propensity for tipping or oscillation during flight, especially in turbulent conditions. Moreover, the lower center of gravity improves control responsiveness, enabling more precise maneuvering and reducing the likelihood of chaotic or unpredictable movements. This stabilization is particularly beneficial during hovering and low-speed flight, contributing to the MAV's overall performance efficiency and stability.

3 FLIGHT DYNAMICS

To demonstrate the effectiveness of the adjustable abdomen control system, it is essential to analyse the dynamic response to abdomen movement. Due to the highly nonlinear nature of the dynamics of FWMAV, the following assumption are made to simplify the model and only retain the most essential and relevant dynamics:

- Longitudinal and lateral dynamics are decoupled and in our model, only the longitudinal dynamics are considered to capture the pitch performance enabled by the abdomen.
- The initial equilibrium condition of the FWMAV is set to be hover.
- The upper and lower body of the FWMAV are modeled as two concentrated masses on massless rigid rods connected by an actuated joint.

3.1 Equation of Motion

The equation of motion is derived using the Lagrange-Euler approach. The dynamic is captured by two pendulums namely, the upper body, and the abdomen body, both of which rotate about the abdomen joint, and the overall system, featuring translational and rotational degree of freedom. The two masses m_1 and m_2 are attached to the ends of two rods of lengths l_1 and l_2 meaning upper body and abdomen respectively. The rods

http://www.imavs.org/

http://www.imavs.org/

are assumed to be massless and rigid, and the system is subject to gravity.

Consider first a 2D problem in the longitudinal direction for a generic FWMAV: the undulating of the body is described by angular displacements θ_1 for the upper body and θ_2 for the abdomen. The aerodynamic forces and moments acts on the quarter point of the thorax. Five degrees of freedom were identified and illustrated in Figure 2

- x: Horizontal position of the joint.
- z: Vertical position of the joint.
- θ_t: Angular displacement of the thorax.
- θ_a: Angular displacement of the abdomen.
- θ_0: Pitch angle of the body with respect to the vertical axis.

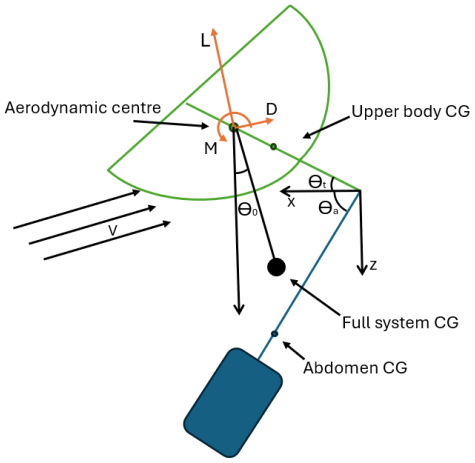


Figure 2: Free body diagram of the FWMAV in a generic condition

3.1.1 Kinetic and Potential Energy

To start with, Euler-Lagrange equations for the system are derived from the Lagrangian $L = T - V$:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_i}\right) - \frac{\partial L}{\partial \theta_i} = 0 \tag{1}$$

The kinetic energy T of the system is:

$$T = \frac{1}{2}m_1\left(\dot{y}_1^2 + \dot{z}_1^2\right) + \frac{1}{2}m_2\left(\dot{y}_2^2 + \dot{z}_2^2\right) \tag{2}$$

The potential energy V includes gravitational potential energy and additional potential energy related to the pitch angle similar to a pendulum hanging from the aerodynamic center. Note that the pitch angle is assumed to be small and therefore small angle approximation applies. This is a valid assumption because we are mostly concerned with small perturbations around the equilibrium point, in this case, hovering:

$$\begin{aligned} V = m_1 g y_1 + m_2 g y_2 = -m_1 g l_1 \cos\theta_1 \\ -m_2 g(l_1\cos\theta_1 + l_2\cos\theta_2) \end{aligned} \tag{3}$$

The Lagrangian L is:

$$L = T - V \tag{4}$$

$$\begin{aligned} L = \frac{1}{2}(m_1+m_2)l_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2 l_2^2\dot{\theta}_2^2 \\ +m_2 l_1 l_2 \dot{\theta}_1\dot{\theta}_2\cos(\theta_1-\theta_2) \\ +(m_1+m_2)g l_1\cos\theta_1 + m_2 g l_2\cos\theta_2 \end{aligned} \tag{5}$$

The Euler-Lagrange equations give us the equations of motion:

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_1}\right) - \frac{\partial L}{\partial \theta_1} = 0, \\ \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_2}\right) - \frac{\partial L}{\partial \theta_2} = 0 \end{aligned} \tag{6}$$

For θ_1:

$$\begin{aligned} (m_1+m_2)l_1^2\ddot{\theta}_1 + m_2 l_1 l_2\ddot{\theta}_2\cos(\theta_1-\theta_2) \\ +m_2 l_1 l_2\dot{\theta}_2^2\sin(\theta_1-\theta_2) \\ +(m_1+m_2)g l_1\sin\theta_1 = 0 \end{aligned} \tag{7}$$

For θ_2:

$$\begin{aligned} m_2 l_2^2\ddot{\theta}_2 + m_2 l_1 l_2\ddot{\theta}_1\cos(\theta_1-\theta_2) \\ +m_2 l_1 l_2\dot{\theta}_1^2\sin(\theta_1-\theta_2) \\ +m_2 g l_2\sin\theta_2 = 0 \end{aligned} \tag{8}$$

3.1.2 Stability Derivatives and System Matrix

For small perturbations around the downward equilibrium ($\theta_1 = 0$, $\theta_2 = 0$), the linearised equations are:

$$\begin{aligned} (m_1+m_2)l_1\delta\ddot{\theta}_1 + m_2 l_2\delta\ddot{\theta}_2 \\ +(m_1+m_2)g\delta\theta_1 = 0 \end{aligned} \tag{9}$$

$$m_2 l_2\delta\ddot{\theta}_2 + m_2 l_1\delta\ddot{\theta}_1 + m_2 g\delta\theta_2 = 0 \tag{10}$$

The linearised system can be written in state-space form as:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} \tag{11}$$

where the state vector $\mathbf{x}$ is:

$$\mathbf{x} = \begin{bmatrix} \theta_1 & \theta_2 & \dot{\theta}_1 & \dot{\theta}_2 \end{bmatrix}^T \tag{12}$$

The system matrix $\mathbf{A}$ is:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{(m_1+m_2)g}{(m_1+m_2)l_1} & 0 & 0 & -\frac{m_2 l_2}{(m_1+m_2)l_1} \\ 0 & -\frac{g}{l_2} & -\frac{l_1}{l_2} & 0 \end{bmatrix}$$

The elements of $\mathbf{A}$ correspond to the stability derivatives Z_{θ_1}, Z_{θ_2},$Z_{\dot{\theta}_1}$,$Z_{\dot{\theta}_2}$. The aerodynamic forces were neglected because this analysis is to simulate a hanging situation of the model subject to disturbance.

To analyse stability, we need the characteristic equation of the system. Assume solutions of the form $\delta\theta_1 = Ae^{\lambda t}$ and $\delta\theta_2 = Be^{\lambda t}$. Substitute these into the linearized equations to find λ:

$$\lambda^2 A = -\frac{m_2 l_2}{(m_1+m_2)l_1}\lambda^2 B - \frac{g}{l_1}A$$

$$\lambda^2 B = -\frac{l_1}{l_2}\lambda^2 A - \frac{g}{l_2}B$$

This yields a system of linear equations that can be expressed as:

$$\begin{bmatrix} \lambda^2 + \frac{g}{l_1} & \frac{m_2 l_2}{(m_1+m_2)l_1}\lambda^2 \\ \frac{l_1}{l_2}\lambda^2 & \lambda^2 + \frac{g}{l_2} \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

The determinant of this system must be zero for nontrivial solutions:

$$\left(\lambda^2 + \frac{g}{l_1}\right)\left(\lambda^2 + \frac{g}{l_2}\right) - \frac{m_2 l_2}{(m_1+m_2)l_1} \cdot \frac{l_1}{l_2}\lambda^4 = 0$$

Simplifying gives:

$$\lambda^4 + \lambda^2\left(\frac{g}{l_1} + \frac{g}{l_2}\right) + \frac{g^2}{l_1 l_2} = 0$$

This is a quadratic equation in λ^2:

$$\lambda^2 = -\frac{1}{2}\left(\frac{g}{l_1} + \frac{g}{l_2}\right) \pm \sqrt{\left(\frac{1}{2}\left(\frac{g}{l_1} + \frac{g}{l_2}\right)\right)^2 - \frac{g^2}{l_1 l_2}}$$

Applying this onto the model gives $\lambda^2 > 0$ and λ is real, indicating an exponential decay and a stable system.

3.1.3 Settling Time After a Disturbance

The natural frequencies of the system are:

$$\omega_1 = \sqrt{\frac{g}{l_1}}, \quad \omega_2 = \sqrt{\frac{g}{l_2}} \tag{13}$$

The system is critically damped if the damping is just sufficient to return the system to equilibrium without oscillations. Adding critical damping to the velocity terms:

$$c_1 = 2\sqrt{(m_1+m_2)gl_1}, \quad c_2 = 2\sqrt{m_2 g l_2} \tag{14}$$

The time constants τ for damping are:

$$\tau_1 = \frac{1}{\omega_1}, \quad \tau_2 = \frac{1}{\omega_2} \tag{15}$$

The settling time T_{settle} is approximately:

$$T_{\text{settle}} = 4 \times \max(\tau_1, \tau_2) \tag{16}$$

For the prototype with $l_1 = 0.32\,\text{m}$, $l_2 = 0.1\,\text{m}$, $m_1 = 0.013\,\text{kg}$, $m_2 = 0.007\,\text{kg}$, and $g = 9.81\,\text{m/s}^2$:

$$\omega_1 = \sqrt{\frac{9.81}{0.32}} \approx 5.54\,\text{rad/s} \tag{17}$$

$$\tau_1 = \frac{1}{5.54} \approx 0.18\,\text{s} \tag{18}$$

$$T_{\text{settle}} \approx 0.72\,\text{s} \tag{19}$$

This settling time gives an estimate of the short time it takes for the model to stabilise after a disturbance, indicating its promising stability property.

4 EXPERIMENTAL AND SIMULATION SETUP

This section introduces the experimental setup that tracks and measures the abdomen movement utilising computer vision tracking techniques and high-speed cameras. The experiment is designed to measure the transient response of the MAV induced by instantaneous abdomen actuation in near hover conditions and thus characterise the pitch angle control by abdomen actuation.

4.1 Physical hardware setup

The dynamics of the MAV was captured using a monocular, high-speed camera system as shown in Figure 3. The hover condition of MAV was assumed to be equivalent to the MAV suspended with a long string, which provides a simulated aerodynamic force under

http://www.imavs.org/

small angle deviation from equilibrium. The string was designed to be long to minimise damping effect. A step control signal of 90 deg actuation was sent out from an Arduino UNO microcontroller. To record the MAV's dynamic behavior, a Phantom Miro high-speed camera was positioned perpendicular to the plane of pitch rotation, operating at a frame rate of 1000 frames per second.

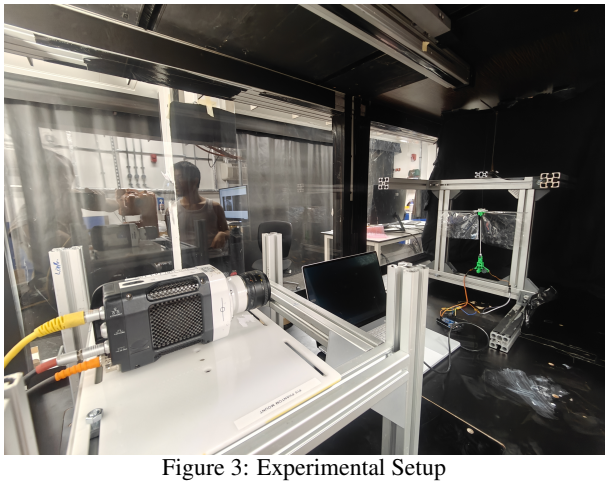


Figure 3: Experimental Setup

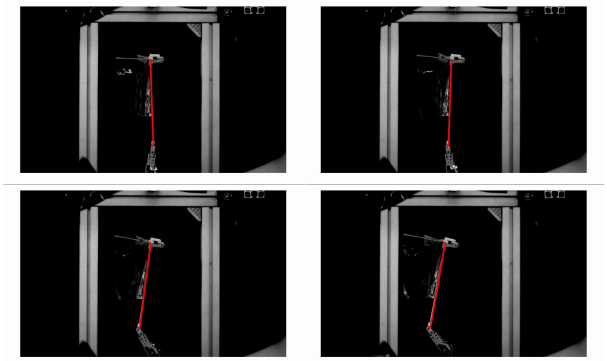


Figure 4: Composite frames taken from high speed footage

4.2 Computer vision powered velocity tracking system

A computer vision algorithm was deployed to process the images and characterize the dynamics of the MAV. The analysis begins by designating the first frame of each video sequence as the reference frame, against which the angular displacement of the MAV's upper body in all subsequent frames is calculated. This method allows for accurate tracking of rotational dynamics over time. Regions of Interest are employed to isolate the MAV's upper body within the video frames to optimize computational efficiency.

Feature tracking was achieved using the MinEigenFeatures method, which detects stable and distinctive points on the MAV's upper body by analyzing the minimum eigenvalue of gradient matrices within the image. These features, identified in the reference frame, are consistently tracked across subsequent frames, enabling precise measurement of the MAV's angular displacement.

To mitigate potential inaccuracies due to video noise and artifacts, the Savitzky-Golay filter is applied as a noise reduction technique. This filter smooths the data while preserving essential features such as peaks and rotational transitions, thereby enhancing the reliability of the calculated angular displacements. Collectively, these computer vision techniques enable precise and reliable characterization of the MAV's dynamic behavior under experimental conditions.

4.2.1 Simulation Setup

Meanwhile, a simulation of the MAV was conducted using the Webots platform to complement the experiment and compared with the experimental results. Instead of using the previously derived simplified model, the simulation was created with the in-built high fidelity physics engine in Webot, capturing the geometrical and dynamical properties more accurately. The simulation environment was set up with all relevant physical properties precisely defined, including the MAV's weight, center of gravity, and detailed physical geometries. These parameters ensure that the virtual model closely mirrors the MAV, providing a reliable foundation for dynamic analysis and will be subsequently used for research in the control algorithm.

http://www.imavs.org/

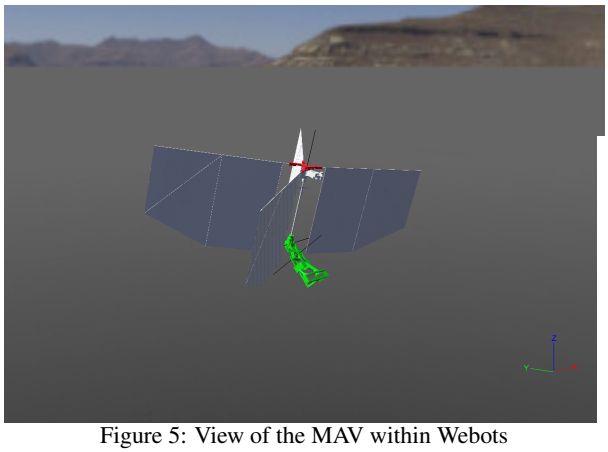


Figure 5: View of the MAV within Webots

Since we are only interested in the near hover condition of the model, the aerodynamic model was simplified and represented by a varying thrust along the direction of the upper body. The unsteady lift in a flapping cycle was setup with experimental data from the wind tunnel testing [13] and was implemented in the model. To achieve near-hover condition in the simulation, the flapping frequency is set to 13Hz. Through iterations, the unsteady lift data was scaled to generate enough thrust to maintain hover condition throughout simulated experiment. The period is set to be 0.0769 seconds as a result of a flapping frequency of 13 Hz accordingly. The aerodynamic centre is assumed to be quarter point of the wing chord and the torque model was derived as shown in the appendix and implemented in the simulation model.

4.3 Results and Discussion

The experiments and simulations conducted in this study provide substantial evidence supporting the effectiveness of the bio-inspired abdomen mechanism in controlling the pitch dynamics of the FWMAV. The results are organized into the following key areas: characterization of the abdomen's impact on pitch control, verification and comparison through simulation.

4.3.1 Characterization of Abdomen Impact on Pitch Control

The experimental setup demonstrated the ability of the abdomen to effectively control the pitch angle of the MAV. By giving an actuation input, we observed a clear and measurable change in the pitch angle, confirming the proposed hypothesis that the abdomen can significantly influence the vehicle's dynamics.

The results are as shown in Figure6: multiple trials with different weight distributions between the body and the abdomen to examine how varying the mass affects the pitch response. The experiments verified the hypothesis that increased abdomen mass or percentage mass over the whole body resulted in a higher pitch angle. This information will be used to inform on the optimal mass allocation minimizing control effort.

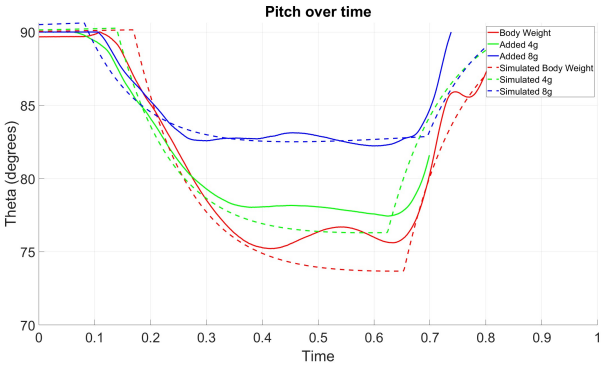


Figure 6: Resulting upper body pitch deflection for Body weight, 4g weight, and 8g weight

4.3.2 Simulation Result Comparison

The simulation results shows consistancy with the experimental data in general validating the simulation model dynamic. As shown in Figure 6, the simulated pitch angles follow the experiment data closely until experimental pitch reached minimum. Then the experimental pitch angle showed increasing rebounce as the abdomen weight increases. This is due to the increasing restoring damping effect of the attached string as the pitch and therefore displacement angle of the string becomes larger, which is what is expected. However, the high degree of correlation between the experimental and simulated results in transient phase in all three cases reinforces the robustness of the simulation model and its potential for further research on the control dynamic of the MAV.

4.3.3 Discussion of Experimental Setup and Assumptions

While the experiments and simulations provided valuable insights, certain assumptions were made in the setup that may affect the generalisability of the results. For instance, the use of a suspended string to simulate hover conditions, while effective, can only be valid in the transient response and unable to capture the full envelope of near hover dynamic. Future studies will need to explore more advanced experimental setups that fully capture the MAV's dynamics in free-flight conditions. Additionally, the linearised equations of motion used in the stability analysis assume small perturbations around the equilibrium point. While this assumption holds for hover conditions, it may not be fully applicable to more dynamic flight scenarios. Further research should explore the full nonlinear dynamics of the MAV

http://www.imavs.org/

to provide a more comprehensive understanding of its stability and control characteristics.

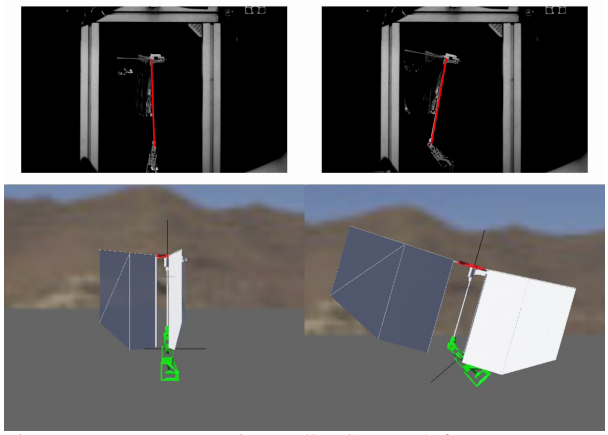


Figure 7: Top: Experimentally Captured frames, Bottom: Simulation

5 CONCLUSION AND FUTURE WORK

This paper introduced a novel flapping wing micro aerial vehicle (FWMAV) with a bio-inspired abdomen mechanism designed to enhance flight stability and maneuverability. Experimental and simulation results confirmed that the abdomen mechanism effectively controls pitch and stabilizes the vehicle by adjusting the center of mass. This bio-inspired approach offers a promising method for achieving agile control in dynamic environments while being passively stable, contributing valuable insights to the design of micro aerial vehicles. Future work will include developing a comprehensive model to explore stability across various flight conditions; conducting experiments in untethered, three-dimensional setups to validate the control mechanisms in realistic scenarios; and using the existing setup to quantify and compare the passive stability as well as maneuverability enabled by abdomen control against the existing control strategies in terms of effectiveness and energy efficiency.

REFERENCES

[1] M. Keennon, K. Klingebiel, and H. Won. Development of the nano hummingbird: A tailless fapping wing micro air vehicle. In *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 0588, Nashville, Tennessee, 2012. AIAA.

[2] C. De Wagter, S. Tijmons, B. D. W. Remes, and G. C. H. E. De Croon. Autonomous fight of a 20-gram fapping wing mav with a 4-gram onboard stereo vision system. In *IEEE International Conference on Robotics and Automation*, pages 4982–4987. IEEE, 2014.

[3] M. Karasek, F. T. Muijres, C. De Wagter, B. D. W. Remes, and G. C. H. E. De Croon. A tailless aerial robotic fapper reveals that fies use torque coupling in rapid banked turns. *Science*, 361:1089, 2018.

[4] Q. V. Nguyen and W. L. Chan. Development and fight performance of a biologically-inspired tailless fapping-wing micro air vehicle with wing stroke plane modulation. *Bioinspiration & Biomimetics*, 14:016015, 2018.

[5] Graham K. Taylor and Adrian L. R. Thomas. Active and passive stabilization of body pitch in insect flight. *Journal of The Royal Society Interface*, 10(88):20130237, 2013.

[6] Mao Sun. Aerodynamic performance and stability of insect-inspired flights with. *TLA*, 195:1–10, 2019.

[7] Wei Shyy, Hikaru Aono, Srinath K. Chimakurthi, and Patrick Trizila. Aerodynamics, sensing and control of insect-scale flapping-wing flight. *Journal of the Royal Society Interface*, 10(86):20130297, 2014.

[8] Mao Sun and Yan Xiong. Dynamic flight stability of hovering insects. *Acta Mechanica Sinica*, 23(6):635–647, 2007.

[9] Jonathan P. Dyhr, Kristi A. Morgansen, Thomas L. Daniel, and Noah J. Cowan. Flexible strategies for flight control: an active role for the abdomen. *Journal of Experimental Biology*, 216(9):1523–1536, 2013.

[10] Noah J. Cowan, Kristi A. Morgansen, Thomas L. Daniel, and Jonathan P. Dyhr. Model predictive control of abdominal movements for flight stability in hovering moths. *IEEE Transactions on Robotics*, 30(4):853–864, 2014.

[11] Ralph A. DiCaprio and Noah J. Cowan. Vibrational control: A hidden stabilization mechanism in insect flight. *Science Robotics*, 5(48):eabb1502, 2020.

[12] Quoc-Viet Nguyen, Woei Leong Chan, and Marco Debiasi. An insect-inspired flapping wing micro air vehicle with double wing clap-fling effects and capability of sustained hovering. In *Bioinspiration, Biomimetics, and Bioreplication 2015, edited by Akhlesh Lakhtakia, Mato Knez, Raúl J. Martín-Palma*, page 94290U. SPIE, 2015.

[13] S. F. Armanini, J. V. Caetano, G. C. H. E. de Croon, C. C. de Visser, and M. Mulder. Quasi-steady aerodynamic model of clap-and-fling flapping mav and validation using free-flight data. In *AIAA*

http://www.imavs.org/

Atmospheric Flight Mechanics Conference. AIAA, 2016.

APPENDIX

.1 Derivation of abdomen torque model

To model the relationship between torque and angular position for the servo-actuated abdomen joint, we define the following system parameters for a typical DC servo motor:

- R: Resistance of the motor windings (Ohms)
- L: Inductance of the motor windings (Henrys)
- K_e: Back EMF constant (V/(rad/s))
- K_t: Torque constant (Nm/A)
- J: Moment of inertia (kg m^2)
- B: Damping coefficient (Nm/(rad/s))

The voltage equation for the motor and the torque produced by the motor are respectively:

$$V(t) = I(t)R + L\frac{dI(t)}{dt} + K_e\omega(t)$$

$$\tau(t) = K_t I(t)$$

$$\tau(t) = J\frac{d\omega(t)}{dt} + B\omega(t)$$

we can easily get:

$$V(t) = \frac{LJ}{K_t}\frac{d^2\omega(t)}{dt^2} + \left(\frac{JR}{K_t} + \frac{LB}{K_t}\right)\frac{d\omega(t)}{dt} + \left(\frac{BR}{K_t} + K_e\right)\omega(t)$$

.1.1 Torque as a Function of Angular Position

The torque function derived earlier is:

$$\tau(t) = \frac{K_t}{R}V(t) - \frac{LJ}{R}\omega''(t) - \left(J + \frac{LB}{R}\right)\omega'(t) - \left(B + \frac{K_e}{R}\right)\omega(t)$$

In the Laplace domain, substituting $V(s) = (as^2 + bs + c)\Theta(s)$ into the torque function and simply we get:

$$\tau(s) = \left(\frac{K_t LJ}{RK_t} - \frac{LJ}{R} - \frac{RJ}{R} - \frac{LB}{R}\right)s^2\Theta(s) + \left(\frac{K_t\left(\frac{JR}{K_t} + \frac{LB}{K_t}\right)}{R} - \frac{B + K_e}{R}\right)s\Theta(s) + \frac{K_t\left(\frac{BR}{K_t} + K_e\right)}{R}\Theta(s)$$

Let:

$$A = \frac{K_t a - LJ - RJ - LB}{R}$$

$$B = \frac{K_t\left(\frac{JR}{K_t} + \frac{LB}{K_t}\right) - B - K_e}{R}$$

$$C = \frac{K_t\left(\frac{BR}{K_t} + K_e\right)}{R}$$

Hence, the torque in terms of the angular position in the Laplace domain is as follow and a transfer function relating torque produced to angular position of the motor given an actuation command is derived.

$$\tau(s) = \left(As^2 + Bs + C\right)\Theta(s) \tag{20}$$

.1.2 Servo Control

A regular servo typically actuates from point A to point B at a constant speed. However, this constant speed can introduce a sudden impulse or jerk, particularly in delicate mechanisms like flapping-wing micro aerial vehicles (FWMAVs). This jerking motion can negatively impact the stability and performance of the MAV. To address this issue, velocity control strategies were implemented to smooth the transition between positions, thereby reducing the jerk and enhancing the overall stability and performance of the MAV.

The goal is to transition the servo from the initial angle α to the objective angle β within a specified actuation time T, in a manner that minimizes jerk. This was achieved by controlling the velocity profile of the servo.

$$a0 = \alpha$$

$$a1 = \frac{3}{T^2}\cdot(\beta - \alpha)$$

$$a2 = -\frac{2}{T^3}\cdot(\beta - \alpha)$$

Where:

- $a0$, $a1$, and $a2$ are coefficients that define the polynomial for the position $\theta(t)$ over time.
- α is the initial angle.
- β is the objective angle.
- T is the total time of actuation.
- t is the current time step.

http://www.imavs.org/

The position $\theta(t)$ of the servo at any given time t is described by the cubic polynomial:

$$\theta(t) = a0 + a1 \cdot t^2 + a2 \cdot t^3$$

This polynomial ensures a smooth transition from α to β over the time T.

1. Initial Conditions:
   (a) At $t = 0$, $\theta(0) = \alpha$. Thus, $a0 = \alpha$.
   (b) At $t = T$, $\theta(T) = \beta$.
2. Velocity Constraints:
   (a) The initial velocity and final velocity is zero, ensuring smooth stop.
3. Jerk minimization:
   (a) The cubic polynomial $\theta(t) = a0 + a1 \cdot t^2 + a2 \cdot t^3$ is chosen to ensure continuous acceleration and to minimize jerk.

http://www.imavs.org/

A method for determining the safe separation to an unknown-model uncooperative UAV in U-space

Ivan Panov*, Jes Jepsen**, Mirko Presser* and Kjeld Jensen**
* Department of Business Development and Technology, Aarhus University, Herning, Denmark 7400
** SDU Drone Center, University of Southern Denmark, Odense, Denmark 5230

http://www.imavs.org/

ABSTRACT

Unmanned aerial vehicles (UAVs), often referred to as drones, usher in a new era of aviation by delivering new services at very low-level airspace. Many technical and organisational issues must be addressed to ensure flight safety at the European level. This paper proposes a novel method for addressing the challenge of ensuring a safe minimum separation distance between a conventional drone (CD) and an uncooperative drone (UD) of an unknown model (UDUM). A CD follows regulations and maintains communication with other U-space traffic participants, while a UD conducts a non-conformance flight without permission from the U-space service provider (USSP), thereby posing a significant threat to flight safety.

Our safe separation method relies on a global UAV database analysis and assumes that a vision system or other passive sensing technology (a recognition system) can identify the type of UAV of an unknown model. Additionally, the method involves determining the minimum separation distance and safety radius of a UDUM using velocity vectors for both the CD and UDUM and performing geometric optimisation.

Our UDUM classification led to a significant improvement in measuring the minimum separation distance, resulting in an up to fivefold optimisation among different types of UAVs. We recommend the method for the future autonomous guidance system of CDs.

1 INTRODUCTION

U-space, the European implementation of an unmanned aircraft system traffic management (UTM) system, is expected to support UAV operations with very high levels of automation [1, 2, 3] including autonomous UAV guidance that relies on the planning of safe 4D trajectories encompassing three dimensions and time [4]. The autonomous system will rely on automated rule-based systems and, potentially, artificial intelligence to plan trajectories and continuously replan [4].

Collision avoidance is a fundamental principle of flight [5]. Through their literature review, J. K. Kuchar et al. made a significant contribution to the field by categorising the existing methods [6]. Some scientific articles have proposed various collision avoidance approaches for CDs [7], [8]. The performance and intentions of UDUM aircraft are unknown; therefore, classical methods such as probabilistic and deterministic approaches cannot be used, as they rely on this data. If U-space airspace is violated by a UDUM, it introduces an elevated risk of potential collision with surrounding aircraft. The fundamental question here is what "surrounding" means; in other words, what is the safe separation distance needed for each case.

In this paper, we discuss various UAV models and types. UAV models are characterized by a unique design or configuration, including specific features such as weight, size, and propulsion system. In contrast, UAV type refers to the physical configuration, operational characteristics, or usage of the UAV. We categorize UAVs into the following types: fixed-wing, rotary-wing, lighter-than-air, flapping-wing, and parafoil. For example, the DJI Phantom 4 is a model, and its type is rotary-wing.

M. Wisniewski et al. [9] proposed to use neural networks to identify a UAV model using its image. The safe separation distance of a known-model UD (UDKM) can be determined if the autonomous guidance system has information about the UDKM's position, aircraft performance and, ideally, its vector of velocity. However, if the UD's model is unknown, finding the minimum safe separation distance becomes significantly more complicated. To the best of our knowledge, there is no known effective solution for determining the minimum safe separation distance of UDUMs. If a UDUM is fast enough to collide with a CD before the flight terminates, and the autonomous guidance system of the CD does not adequately account for this situation, there is scope for flight safety improvements.

We argue that even if the recognition system is unable to identify the model of UD, at least it must be able to identify the UDUM's type based on its image and/or its current speed. Knowing the type of UDUM is essential for calculating the minimum safe separation distance between a CD and a UD. The simple alternative solution of having a standard safe separation distance for all types of UDUMs leads to inefficient

*Email address: ivan.panov@btech.au.dk

U-space airspace usage. This can be especially important for high-density unmanned air traffic.

To tackle the problem, we decided to approach the issue from a new angle. More specifically, we analysed the global database of UAVs [10], which contains information about the maximum speed in horizontal flight of almost 3,000 UAVs. One of our research objectives was to find the maximum speed of each type of UAV.

We start the paper by explaining the assumptions and the research method. After that, we discuss the results of the global database [10] analysis. Then we present a method for using drone-type classification to ensure safe minimum separation. Subsequently, we propose a geometric approach to determining an optimal margin for safe separation. To give an example of how it can be used, we solve a task involving a case scenario of a potential collision. We also discuss whether our method can be used for other cases. Finally, we present a conclusion in the last section.

2 PROBLEM FORMULATION AND RESEARCH METHOD

We advance the hypothesis that if a UD's model is unknown, it is essential to know the maximum speed of its type, the minimum and maximum value of drag coefficient for the type, the aircraft dimensions to predict its cross-sectional area and the maximum and minimum mass range. Based on this information, we propose a method to suggest a safe separation distance between UDUMs and conventional U-space air traffic. The research question is how to find the minimum separation distance and safety radius of UDUMs to prevent collisions with CDs. A safety bubble will likely have a non-spherical form (Figure 8). However, in our simplified approach, the safety bubble is created by calculating the longest distance possible to fly in a predetermined value of time (Figure 8). The points obtained from the calculations are symmetric along the vertical axis, which allows us to discuss minimum separation distance and safety radius on the horizon at a certain altitude (Figure 9).

2.1 Assumptions

Based on the progress of machine learning techniques, we employ the assumption that an artificial intelligence system will be able to identify the type of any drone based on its image [9], [11].

We also make the decision to exclude the fastest highly advanced UAVs from the database, as it is highly unlikely that their model would be unknown. High-speed flight performance requires expensive technologies like jet or turbo engines [12]. Building a compound helicopter model involves very complicated [13] research and development processes, which sets a high bar for making this type of UAV. However, a simple quadcopter can be constructed in a garage or in a drone club by a layman [14]. In the latter case, the UAV's model can be unknown to the drone recognition system, and the CD's autonomous guidance system will not be able to predict a safe separation distance to the UDUM since its performance is unknown. One could argue that certain drones and rockets can fly at supersonic or even hypersonic speeds. However, once again, these technologies are not widely available, and safe separation from them should not be an operational issue for U-space airspace. Protection against such threats is related to defensive technologies and remains outside the scope of the current study.

We propose a solution for 4D (3D + time) space based on the global UAV database, where we categorise various types of UAVs and determine the maximum horizontal flight speed for each type. However, our database is limited in terms of aircraft model, type, and maximum speed, leading us to rely on assumptions concerning drag coefficient, cross-sectional area and mass of the UDUM. We acknowledge that further research is needed to find dependencies between the dimensions of a UAV and its maximum and minimum mass, cross-sectional area and drag coefficient. Based on the data on the UDUM's dimensions and type, the autonomous guidance system can calculate the safe separation distance using our method.

The limitations by type could help achieve collision-free flight in the vicinity of UDUMs operating at different altitudes. Aerodynamic drag, thrust, and mass significantly impact aircraft acceleration, deceleration, and maximum dive speed. While fixed-wing or rotary-wing UAVs may have lower drag coefficient values, some UAVs classified as airships have a large cross-sectional area, which leads to a higher value of the drag coefficient [15]. The greater the drag value, the more severe the limit of maximum diving speed due to the balance of aerodynamic forces [16].

Finally, we did not take into consideration the impact of weather conditions and other specific constraints [17] like restricted areas, sensor accuracy impact, etc. We also chose to disregard the air compression effect, as the increase in drag coefficient with rising speed does not have a detrimental impact on flight safety. Also, the drones we focus on rarely fly significantly faster than 100 m/s. The fastest UAV in our refined dataset has a maximum speed of 118 m/s, and at this velocity, the impact of the effect is relatively small. To fly significantly faster, the drones must dive with a large negative flight path angle γ, which they can only do within a short time interval.

2.2 Research method

In accordance with our assumptions, we refined the global UAV database by excluding the fastest highly advanced UAVs. Subsequently, we checked the maximum speed for each drone type. We developed a novel method, MATHryoshka, that uses data on maximum velocity, drag coefficient, cross-sectional area, mass, and current velocity of UDUM to calculate minimum separation distance and safety radius. To minimise the risk of collisions, we proposed an approach that involves measuring the safe separation distance.

http://www.imavs.org/

http://www.imavs.org/

3 GLOBAL UAV DATABASE ANALYSIS

3.1 Fixed-wing type

To shed light on the maximum speed of basic fixed-wing UAVs, we analysed 1,743 models of this type [10] (Figure 1). We also included tilt-wing UAVs in this list. Some UAVs have a tilt rotor or rotary wing alongside a wing. However, the maximum speed is reached when the rotor generates thrust while the wing generates lift force. This fact allows us to categorise these tilt-wing and tilt-rotor (with wings) UAVs in the same group, namely the fixed-wing type.

101 models of the most technically advanced UAVs were removed from the list, among them the fastest UAVs with jet, turboprop, and rocket engines [12]. The fastest model based on relatively widely accessible technology is the Berkut ISR with a piston engine. Its maximum speed is 425 km/h.

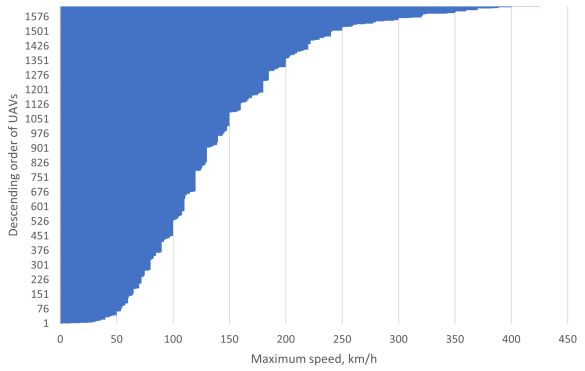


Figure 1: Distribution of the maximum speed of fixed-wing UAVs.

3.2 Rotary-wing type

We analysed and ranked 1,141 rotary-wing UAVs by maximum speed. The 5 models that we defined as highly advanced and fast were excluded from the analysis. Among these were three high-speed compound UAV helicopters: the X2, the HADA and the Jueying-8. We also excluded the Pop.Up Next and the Transporter with turbocharged engines.

After cleaning the data, we built a figure showing the distribution of the rotary-wing UAVs (Figure 2). The figure shows that the majority of the rotary-wing UAVs have a maximum speed ranging from 50 to 150 km/h. However, a Transporter UAV made by Advanced Tactics features a set of widely available technologies: a quadcopter airframe, four petroleum-powered engines, and primitive flight aerodynamics optimisation for the reduction of drag. Its maximum speed is 321 km/h, which is the maximum speed for the rotary-wing type.

3.3 Lighter-than-air type

The lighter-than-air type is exemplified by airships, which tend to have the capability to fly at higher altitudes than rotary-wing and sometimes fixed-wing aircraft. Because of the low air density at altitudes higher than 4,000–6,000 metres, an aircraft can reach its maximum speed. Our study was conducted to assist U-space. Ideally, it is therefore essential to study airships' maximum speeds at a very low airspace level. Since we do not have access to such data, we estimate the airships' maximum speeds with a significant margin of safety. This approach does not affect flight safety; however, we recognise that there is room for further optimisation.

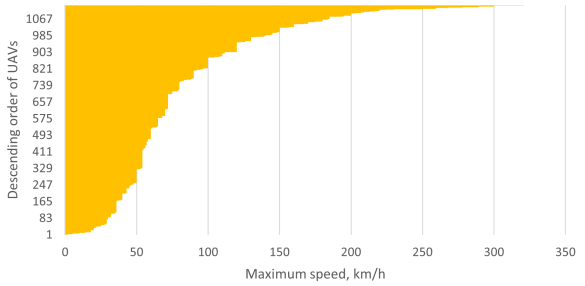


Figure 2: Distribution of the maximum speed of rotary-wing UAVs.

The available database of airships is not large; it only includes 27 models, the fastest of which can reach 150 km/h (Figure 3).

Not powered by engines or motors, hot air balloons are typically not classified as UAVs and are thus not included in our study.

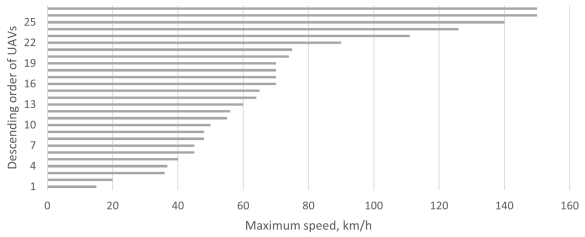


Figure 3: Distribution of the maximum speed of lighter-than-air UAVs.

3.4 Flapping-wing type

Flapping-wing UAVs are among the most rare. In our database, we only have six models, with the fastest one being capable of reaching 125 km/h (Figure 4).

3.5 Parafoil type

Parafoil UAVs, also known as powered parachutes, have a unique design that combines the capabilities of a parachute and propeller-driven aircraft. The lift force is generated by a flexible wing, while thrust is delivered by a propeller-based propulsion system. This type of drone offers a unique mix of payload capacity, stability, and ease of operation. Parafoil UAVs are the slowest of all UAV types; their maximum speed is 80 km/h (Figure 5).

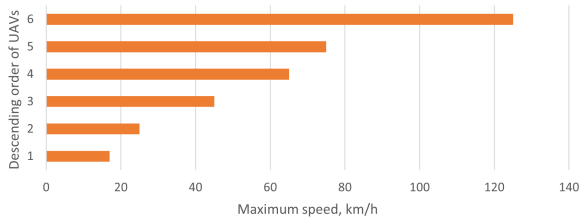


Figure 4: Distribution of the maximum speed of flapping-wing UAVs.

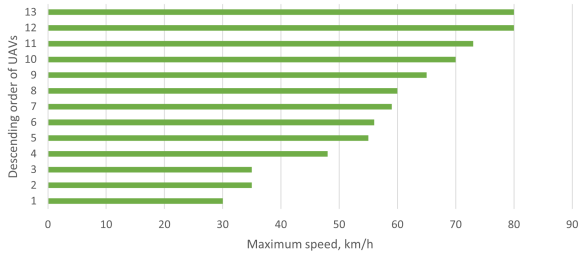


Figure 5: Distribution of the maximum speed of parafoil UAVs.

3.6 *Comparison between types*

When comparing the various types of UAVs, we see a significant difference in their maximum speeds (Figure 6). The fixed-wing aircraft, represented by the fastest type, has a maximum speed of more than 400 km/h. Close to this is the fastest rotary-wing UAV, which can fly faster than 300 km/h. Lighter-than-air and flapping-wing UAVs are relatively slow, with speeds ranging up to 150 km/h. Finally, parafoil UAVs are the slowest, with speeds not exceeding 100 km/h. The most common speeds of rotary-wing and fixed-wing UAVs differ by 100 km/h.

The fastest and the slowest types differ up to five times. It means that the classification proposed can lead to up to 5 five-fold optimisations in identifying UDUM maximum speed. Other, more rare types can achieve maximum speeds that are a couple of times slower than the Berkut.

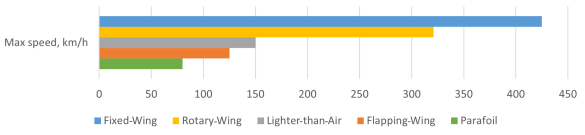


Figure 6: Comparison between types of UAVs.

Based on the global database analysis, we are able to classify a UDUM's maximum speed according to its type. In the next section, we present a mathematical model for a 4D case.

4 MATHEMATICAL MODEL

We calculate the separation distance based on a worst-case scenario to minimise the risk of collision with a known-type UDUM. In our worst-case scenario, we assume that the UDUM is the fastest model of its type and that its trajectory is optimised to collide with an ordinary U-space user. In other words, if the autonomous guidance system can plan U-space air traffic based on the worst-case scenario, the chances of a collision with a UDUM can be reduced significantly.

A collision is only possible if two UAVs have violated the minimum separation distance. To prevent the violation, it is essential to plan the CD's trajectory according to the maximum possible speed of the UDUM, which is affected by the possible range of its mass, its drag coefficient, its cross-sectional area and the lift coefficient for winged UAVs. We only analysed a global UAV database with types of UAVs and their maximum speeds (Figure 6). To predict the range of mass, drag coefficient (C_d), lift coefficient (C_l), and cross-sectional area, a global database with such data and aircraft dimensions is needed. To the best of our knowledge, such a database does not exist. Bearing this in mind, we make calculations using assumptions for the missing data.

If two UAVs are approaching the same coordinates at the same time, there is a potential risk of collision. In this light, we can measure the separation in seconds until the moment of a potential collision. A small adjustment is needed to tackle the problem of aircraft dimensions; an issue we address in the code [18] by adding two aircraft maximum dimensions and dividing the result by two. The deviations here are not very important, since we measure separation distances from the UDUM in the hundreds of metres, while UAV dimensions are only a few to several metres.

There is a significant challenge in making trajectory planning decisions for a CD when the estimated time to potential collision is several minutes. For example, a fixed-wing UAV with a maximum speed of 425 km/h can cover 212.5 km in half an hour. If a UDUM appears, will this mean that the USSP should close this radius of the U-space airspace according to the time of potential approach? As this seems unrealistic, we propose an alternative approach by measuring the time of flight termination for the CD.

4.1 *Minimum separation distance*

If the distance between the UDUM and the CD becomes less than the minimum separation distance, the CD enters an emergency state and must immediately abort the flight. If the CD is equipped with a parachute recovery system, the flight must be terminated and the time of the flight termination is a function of the height above ground level (AGL) and the parachute deployment time. If the CD has no parachute recovery system, it must conduct an emergency landing for which the time is a function of the current height above ground level (AGL), the maximum landing speed and the distance to a safe landing location.

http://www.imavs.org/

http://www.imavs.org/

Since the intentions and aircraft performance of UDUM remain unknown, CD cannot rely solely on superior manoeuvrability to avoid a collision with dodging manoeuvres. If UDUM is more manoeuvrable and possesses a total mechanical energy advantage over CD, there can be no guarantee of collision-free flight if UDUM's intentions are malicious.

Having innovated the MATHryoshka method, we now propose it to find the minimum separation distance, D_{ms}, between a CD and a UDUM. MATHryoshka calculates the distance to a potential collision based on flight time termination.

4.2 MATHryoshka method

The MATHryoshka method relies on a simplified, physical mass-point model of a UDUM moving at various angles γ within the range [-π/2, π/2]. At the start of the calculation, the UDUM activates maximum thrust, representing the worst-case scenario conditions.

Imagine that we have a scenario with a UDUM rotary-wing type. From our previous analysis, we know that its maximum potential horizontal flight speed is 89 m/s. Its current velocity is 10 m/s, which is detected by sensors. Its mass, cross-sectional area, and drag coefficient are given. Information on air density can be collected from U-space; in our case, we assume that it is 1.29 kg/m^3, which is a typical value at sea level.

Given

- UDUM type = rotary-wing.
- Maximum dimension of the UDUM, including width, length and height = 1.5 m.
- Maximum dimension of the CD, including width, length and height = 2 m.
- v_0 = 10 m/s.
- v_{mh} = 89 m/s.
- C_d = 0.2 - drag coefficient for a rotor-wing.
- ρ = 1.225 - air density in kg/m^3 (typical value at sea level).
- A = 0.4 - cross-sectional area, m^2.
- m = 15 kg.
- t = 10 seconds.
- e = 2.71828.
- g = 9.81 m/s2.

Task

Find the distance travelled (s) by UDUM at various γ in the range [-π/2, π/2] within ten seconds, adding the maximum dimensions of the two aircraft divided by 2. The term 'aircraft maximum dimensions' refers to the highest value among the aircraft's height, width, and length.

Solution using the MATHryoshka method

Figure 7 illustrates the vector of thrust and the primary aerodynamic forces acting on UDUM.

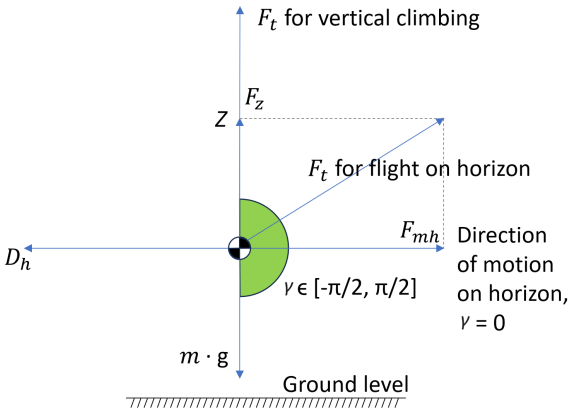


Figure 7: Vector of thrust and the primary aerodynamic forces.

- D_h – force of drag in horizontal flight with maximum thrust.
- F_{mh} – projection of maximum force of thrust on the horizontal axis X in horizontal flight.
- F_z – projection of maximum force of thrust on the vertical axis Z in horizontal flight.
- γ – flight path angle $x \in$ [-π/2, π/2].
- v_0 – initial velocity.
- m – mass.
- g – gravity.
- F_t - maximum force of thrust.

We use the next model simplifications:

- C_d is constant for any γ.
- C_d is constant at any velocity.
- Moments are ignored.

http://www.imavs.org/

According to the Newton's law:

$$F = m \cdot a. \tag{1}$$

Where F = total force, and a = acceleration. Now, we introduce a constant b to simplify the calculus (2).

$$b = \frac{1}{2} \cdot \rho \cdot A \cdot C_d. \tag{2}$$

Where ρ = air density, A = cross-sectional area and C_d = drag coefficient. F_{mh} is constant since the UAV flies with maximum thrust in horizontal flight (3).

$$F_{mh} = b \cdot V_{mh}^2. \tag{3}$$

For horizontal flight, F_z is equal to absolute value of $m \cdot g$, consequently:

$$F_t = \sqrt{F_{mh}^2 + (m \cdot g)^2}. \tag{4}$$

The vector of velocity of UDUM can be aligned with F_t only in vertical diving or climbing. Total force generated by rotary-wing propellers acting in the direction of flight can be estimated in the following way. F_{dm} reflects thrust impact in the direction of flight. It is changing from F_{mh} (for horizontal flight) to F_t (for vertical flight). Consequently, it can be expressed as hypotenuse for both F_{mh} and F_t, depending on the respective γ values.

$$F_{dm} = \sqrt{(F_t \cdot \sin\gamma)^2 + (F_{mh} \cdot \cos\gamma)^2}. \tag{5}$$

F represents the main set of forces acting on UDUM at various flight path angles γ. As inertia and moments are not considered, F always acts in the direction of aircraft trajectory.

$$F = F_{dm} - m \cdot g \cdot \sin\gamma - b \cdot v^2. \tag{6}$$

Using the Euler method, we find the velocity and distance travelled, where s represents the distance travelled in time t at a certain angle γ (9).

$$v_{n+1} = v_n + \frac{F_{dm} - m \cdot g \cdot \sin\gamma - b \cdot v_n^2}{m} \cdot dt. \tag{7}$$

$$t_{n+1} = t_n + dt. \tag{8}$$

$$s_{n+1} = s_n + v_{n+1} \cdot dt. \tag{9}$$

To arrive at a practical solution, the next code was written in Python version 3.10.4 – see [18] for the code link. Using this code, we built Figure 8.

In the centre of the diagram, there is the UAV's centre of mass. According to various angles γ, the UDUM can travel a different distance in time t. Time t is equal to the flight termination time for the CD (we assumed it to be 10 seconds).

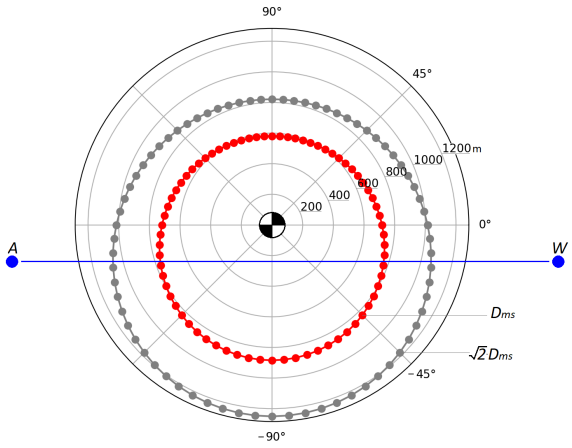


Figure 8: Distance travelled in 10 seconds for each γ, adding the maximum dimensions of the two aircraft divided by 2. Side view.

Point A represents the CD's current position, and W is the destination point. Red points represent the minimum separation, D_{ms}, which is the distance travelled by UDUM at various γ in ten seconds, plus two times the aircraft's maximum dimensions divided by 2 (see the code [18]). We consider the impact of aircraft dimensions since it is essential to mitigate the collision threat.

The grey points represent the safety radius, which is D_{ms} multiplied by $\sqrt{2}$ (13), as we explain in the next section.

4.3 Minimum separation distance and safety area

One of the key issues in avoiding collisions with UDUMs is the uncertainty regarding their intentions and the lack of information about aircraft performance, including unknown limitations on flight characteristics. The current vector of velocity of a UDUM is essential for tactical deconfliction. Also, the data on UAV performance should be collected in a global UAV database, which, as far as we are aware, does not exist yet. The current vector of velocity is not very reliable since it can change depending on different accelerations and dynamics due to the huge diversity among UAV flight characteristics [19]. In light of these limitations, we propose a multifactor approach that incorporates UDUMs' maximum speed, mass, drag coefficient, cross-sectional area, current vector of velocity and geometric optimality.

Consider a situation where a CD intends to fly from point A to a destination, point W (Figure 9). For simplicity's sake, we assume that the CD and the UDUM are at the same altitude. The direct flightpath of the CD intersects with the safety area around a UDUM that has a radius denoted as r (10). The position of the UDUM is known and presented in point I (Figure 9).

$$r = \frac{D_{ms}}{\cos\sigma}. \tag{10}$$

http://www.imavs.org/

A safety area with radius r around point I is essential to ensure an optimal distance between the CD and D_{ms} (Figure 9) at the moment of approach. The optimal distance is the safety area with radius r, which is equal to the IG segment. There is a fundamental problem in determining the value of r. From one perspective, it must be long enough to maximise the right angle σ – the greater the angle, the less the chance that a new random vector of velocity of the UDUM will be threatening the CD. From another perspective, IG must be sufficiently short to reduce the travelling distance of the CD. Measuring G as a point that the CD can reach faster than the UDUM, based on the current vectors of velocity, is not reliable since the UDUM may have the potential to accelerate or decelerate. If we rely on the UDUM's maximum speed, which is higher than that of the CD, there are instances where point G cannot be found at all. Even employing the standard separation distance between CDs defined by regulation cannot guarantee collision avoidance since the intention of the UDUM is unknown and may even be malicious.

Consequently, we propose to use a geometric optimality logic to find the r value. There are two values that are interconnected (10) – angle σ and value r, since lines $u_1 \perp s$ and $ID = IF = IC = D_{ms}$. Also, $s_1 \parallel s$, $u \parallel u_1$, $C \cap s_1$, $C \cap u$, $F \cap t$, $D \cap u_1$, $D \cap s$, $G \cap n_1$, $G \cap t$, $G \cap s_1$, $G \cap u_1$.

The greater angle σ is, the less potential of collision since the intentions of the UDUM are unknown. Conversely, the greater the value of r, the more U-space airspace is covered by the safety area and the longer distance the UDUM will have to cover to come to W through G. For the sake of optimality, it is essential to determine a value of σ that maximises the ratio of σ to r. In that pursuit, we should find a derivative of r and a value of σ that maximises the ratio.

$$\frac{d}{d\sigma}\left(\frac{D_{ms}}{\cos\sigma}\right) = \frac{D_{ms}\cdot\sin\sigma}{\cos^2\sigma}. \tag{11}$$

$$\frac{D_{ms}\cdot\sin\sigma}{\cos^2\sigma} = D_{ms}\cdot\frac{1}{\cos\sigma}. \tag{12}$$

In any case, $\sigma \in (0, \frac{\pi}{2})$, thus the equation (12) is true when $\sigma = \frac{\pi}{4}$, thus $\frac{1}{\cos\sigma} = \sqrt{2}$, which gives the equation (13).

$$r = \sqrt{2}\cdot D_{ms}. \tag{13}$$

The geometric approach may be subject to criticism because it assumes that changing σ is similarly important for trajectory planning as changing the r value. In fact, both are essential, though probably not equally so. We acknowledge the potential for further optimisation in this area.

We disregard an error factor that could be essential for addressing delays associated with data collection, computation, and transfer. The error factor may potentially increase the safety radius required.

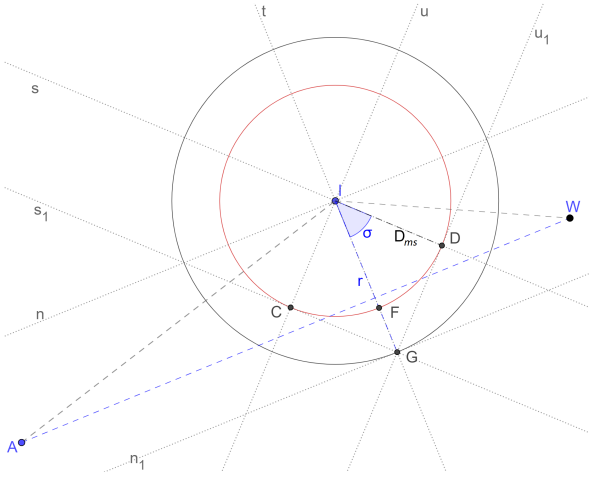


Figure 9: Minimum separation distance and safety radius for γ = 0°. Above view.

In this section, we analysed the scenario where the CD and the UDUM are at the same altitude, which for the worst-case scenario is γ = 0°. However, in actual airspace, altitudes are likely to differ, for example when the altitude of a UDUM is higher than that of a CD (Figure 10). Nevertheless, following the same logic, we can multiply D_{ms} by $\sqrt{2}$ to find the safety radius for each value of γ (Figure 8). For example, for γ = -45°, D_{ms} = 833.5m and r = 833.5m $\cdot\sqrt{2} \approx$ 1179m.

4.4 Some typical scenarios

First, it is essential to identify if the UDUM moves to the location of the CD with the risk of penetrating the safety area. This means that the algorithm of the autonomous guidance system must check if the projection of the vector of velocity of the UDUM, **u**, belongs to the angles ρ_1 or ρ_2 (Figure 10). If **u** belongs to ρ_1 and not to ρ_2, we suggest to avoid collision by choosing a flight path that is closer to point K than point T.

To simplify calculations, we moved the X and Y axes in a way such that (see Figure 11):

- Point $A \in Oy$. Vector **u** $\parallel$ axis Ox. For 3D airspace, **u** is a vector projection of the vector of velocity of the UDUM on the horizontal plane with points OAW.
- Points L, R, $N \in Ox$.
- Ox is a tangent line to the safety circle (with radius r) in point N.
- q and t_1 are the tangent lines to the safety circle with radius r.
- $t_1 \parallel t_2$.

If **u** $\in \rho_2$ and **u** $\notin t_2$ (Figure 11), the waypoint L is a special point that allows the CD to effectively leave the

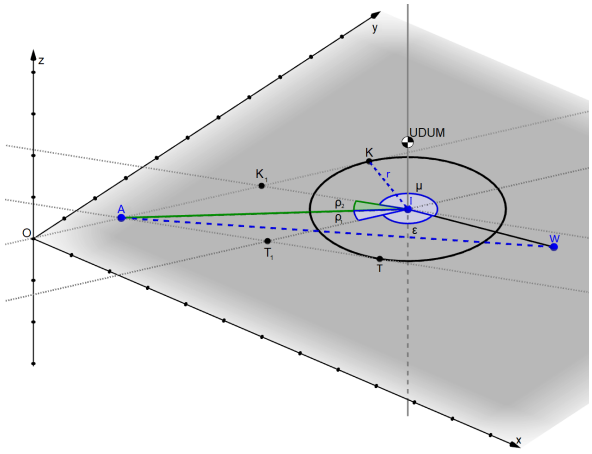


Figure 10: Front collision hazard, UDUM is higher than CD. Two o'clock view angle.

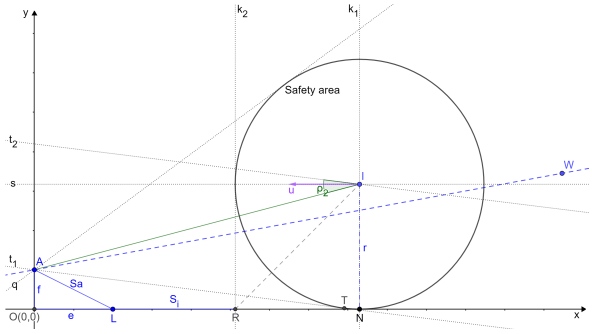


Figure 11: Rho two problem. Above view.

UDUM's direct flightpath, thereby avoiding the safety area of the UDUM. Effectively means that if the two aircraft should move with the same vector of velocity, the CD will reach point L. At the same time, point R will have the same coordinates as L since the UDUM moves in the same direction. Upon reaching point L, the CD is not at risk of violating the minimum separation distance if the UDUM continues its movement in the same direction and the CD continues its flight to point N. In the next section, we provide a solution for finding the coordinates of point L.

4.5 Rho two problem

Task conditions. Given

- $A(0, y_a)$ – location of the CD.
- $f = y_a$.
- $I(x_i, y_i)$ – projection of location of the UDUM on the horizontal plane OAW.
- $W(x_w, y_w)$ – location of the destination point of the CD.
- r – the safety radius (13).
- $\mathbf{u}$ – a vector projection of the vector of velocity of the UDUM on the horizontal plane OAW.
- V_i – the projection of the velocity of the UDUM on plane OAW.
- V_a – the velocity of the CD.
- The CD and the UDUM are in simultaneous flight.
- t_1, q – the tangent lines between point A and the safety circle with centre I and radius r.
- $t1 \parallel t2$.
- ρ_2 – the angle between line AI and line t_2.
- μ - the exterior angle at the points K_1, I, W.
- ϵ - the interior angle at the points T_1, I, W.
- $\mathbf{u} \in \rho_2$ and $\mathbf{u} \notin t_2$.
- Line $OLRN \parallel \mathbf{u}$.
- Line $OLRN$ is a tangent line to the circle with radius r and centre I.
- $RN = IN = r$.

Task

Find x coordinates of point L.

Solution

$$x_r = x_i - r. \tag{14}$$

S_a is the travelling distance of the CD from point A to point L, $OR = j, OL = e, AL = S_a$.

$$e = \sqrt{S_a^2 - f^2}. \tag{15}$$

Since V_i and V_a are known, its ratio k is a constant. S_a is a distance AL, and S_i is a distance RL.

$$\frac{V_i}{V_a} = \frac{S_i}{S_a} = k. \tag{16}$$

$$j - kS_a = \sqrt{S_a^2 - f^2}. \tag{17}$$

$$(k^2 - 1)S_a^2 - 2jkS_a + (f^2 + j^2). \tag{18}$$

$$x_l = e = \sqrt{S_a^2 - f^2}. \tag{19}$$

For $S_a > f$, $V_i > 0$. If $V_a \neq V_i$, $k \neq 1$, S_a and S_i can be intersected at two points on the line x, thus:

$$S_a = \frac{2jk \pm \sqrt{(-2jk)^2 - 4(k^2 - 1)(f^2 + j^2)}}{2(k^2 - 1)}. \tag{20}$$

http://www.imavs.org/

http://www.imavs.org/

However, for the sake of optimality, the smallest positive value of x_l is needed, thus:

$$x_l = \sqrt{\left(\frac{2jk - \sqrt{(-2jk)^2 - 4(k^2-1)(f^2+j^2)}}{2(k^2-1)}\right)^2 - f^2}. \tag{21}$$

If $V_i > 0$ and $f > S_a$, the CD is at risk of penetrating the UDUM's safety area wherever the CD flies. Mathematically, this means that the discriminant cannot be negative. It means that solutions exist for:

$$k < \sqrt{1 + \left(\frac{j}{f}\right)^2}. \tag{22}$$

If $V_a = V_i$, thus $S_a = S_i$, and $k = 1$, thus:

$$(j - S_i)^2 = S_i^2 - f^2. \tag{23}$$

$$S_i = \frac{j^2 + f^2}{2j}. \tag{24}$$

$$x_l = j - \frac{j^2 + f^2}{2j}. \tag{25}$$

If $V_i > 0$ and $f = S_a$, thus:

$$x_l = 0. \tag{26}$$

If $V_i > 0$, then vector $\mathbf{u}$ = 0, consequently, point R cannot be defined. We suggest heading to point T in this case; however, that is a different task.

If $\mathbf{u} \in \rho_1$ and $\mathbf{u} \notin \rho_2$ and $\mathbf{u} \notin \varepsilon$ (Figure 10), and AW intersects with the safety area, the task can be solved in a similar way, but by ensuring that the CD heads to the other (right) side of the UDUM.

We assume that the velocity vector of the UDUM and its projection $\mathbf{u}$ are known in real time. Therefore, recalculating point L every few milliseconds (for example) could move its coordinates to a new safe position if the UDUM accelerates. Point L can also be moved to a more optimal position if the UDUM decelerates during the flight.

If $\mathbf{u} \in \mu$ or $\mathbf{u} \in \varepsilon$ and AW intersect with the safety area, other challenges arise in which the minimum separation distance D_{ms} and the safety radius r play significant roles in defining the problem and finding a suitable solution. Another example is when AW does not intersect with the safety area (Figure 12), but there is a potential risk that the minimum separation distance may be breached if the UDUM is fast enough. A_1 is the position of the CD, and W_1 is the position of the UDUM. A_1T_2 and W_1T_1 are tangent lines to the circle of the safety area. $A_2I_1 \parallel A_1T_2$, $T_1W_1 \parallel I_1W_2$. $\mathbf{u}_1 \in \rho_3$.

To solve such a task, we need to know the minimum separation distance, D_{ms}, and the safety radius, r.

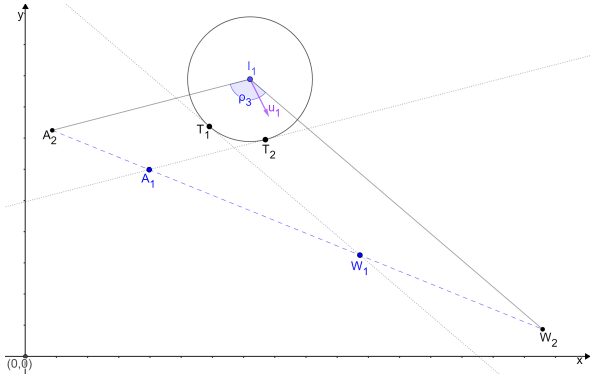


Figure 12: Trajectory intersection in case of remote positions. Above view.

5 CONCLUSION

This article proposes a novel method for calculating the minimum separation distance and the safety radius for a UDUM whose type was identified by an image. The method includes global UAV database analysis, classification of the existing drones, determination of the fastest UAVs among types, and a mathematical model for estimating the safety radius and the minimum separation distance. To make the method available for the autonomous guidance system, we suggest creating a global UAV database that would allow us to predict the range of drag coefficient, lift coefficient (for UAVs with wings), mass and cross-sectional area based on UAV dimensions.

Relying on the method and the assumptions made, we solved the challenge of collision avoidance when a UDUM approaches a CD, posing a risk of infringement of the minimum separation distance. We also discussed the typical collision hazard patterns for which the method can be used in future research. To quantify the method, we suggest performing the appropriate experiments in a simulated environment.

The continuous development of UAV technologies underscores the importance of maintaining an updated global UAV database. Such an approach could assist in measuring minimum separation distances by using the latest advancements in widely available modern technologies. In this light, the study on UDUM type identification by image or video looks promising.

The findings show that the suggested method allows us to deal with UDUMs whose types were identified by the recognition system. The recommended separation distances measured in hundreds of metres appear realistic and feasible. The significant differences in the maximum speeds among the various types of UAVs reflect the optimisation of the problem.

In case the recognition system cannot identify the UDUM's type, the UDUM can be classified as the fastest type to guarantee flight safety.

REFERENCES

[1] The European Commission. Commission implementing regulation (eu) 2021/664. volume 64, page 161–183. Official Journal of the European Union, 2021.

[2] CORUS-XUAM. U-space ConOps and architecture (edition 4, 01.00.02). pages 1–116. CORUS-XUAM, 2023.

[3] JARUS. JARUS Methodology for Evaluation of Automation for UAS Operations, Joint Authorities for Rulemaking of Unmanned Systems. pages 1–44, 2023.

[4] S. Ramasamy A. Gardi and R. Sabatini. 4-dimensional trajectory generation algorithms for RPAS Mission Management Systems. page 386–394, 2015.

[5] SKYbrary Aviation Safety. Collision avoidance. Electronic repository of safety knowledge SKYbrary, 2023.

[6] L. C. Yang J. K. Kuchar. A review of conflict detection and resolution modeling methods. 1(4):179–189, 2000.

[7] D. Ghose L. A. Tony and A. Chakravarthy. Avoidance maps: A new concept in uav collision avoidance. page 1483–1492, 2017.

[8] G. Mengali G. Migliaccio and R. Galatolo. Conflict detection and resolution algorithms for uavs collision avoidance. 118(1205):828–842, 2014.

[9] Z. A. Rana M. Wisniewski and I. Petrunin. Drone model classification using convolutional neural network trained on synthetic data. 8(8), 2022.

[10] P. van Blyenburgh. Rpas remotely piloted aircraft systems the global perspective - volume i - 18th edition. pages 1–214. Blyenburgh and Co, 2023.

[11] A. Géron. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd ed.* O'Reilly Media, Incorporated, 2019.

[12] F. Zhao B. Zhang, Z. Song and C. Liu. Overview of propulsion systems for unmanned aerial vehicles. 15:1–25, 2022.

[13] A. Duran G. Heredia and A. Ollero. Modeling and simulation of the hada reconfigurable uav. 65(1):115–122, 2012.

[14] M. K. Orosz G. Korsoveczki and I. Balajti. Performance analysis of quadcopter drones to radar detection and track initialization characteristics optimization. page 1–10, 2023.

[15] A. Forrester A. Sobester. *Aircraft Aerodynamic Design: Geometry and Optimization*. John Wiley and Sons, Ltd, 2014.

[16] E. Torenbeek H. Wittenberg. *Flight Physics, 1st ed.* Springer Science+Business Media B.V., 2009.

[17] I. Panov and A. Ul Haq. A critical review of information provision for U-space traffic autonomous guidance. *MDPI Aerospace*, page 471, 2024.

[18] I. Panov. Model of separation with UDUM rotary-wing aircraft via MATHryoshka method (python version 3.10.4), https://github.com/SkyIvanCoding/MATHryoshka_method. *Github, may. 05, 2024*.

[19] M. Sadraey. *Unmanned Aircraft Design: A Review of Fundamentals*, volume 1. Synthesis Lectures on Mechanical Engineering, 2017.

http://www.imavs.org/

A Detect And Avoid concept for UAS operations near Hospital Emergency Medical Services helipads in uncontrolled urban airspace (DRAFT VERSION).

Kjeld Jensen[1], Klavs Andersen[2] and Jes Hundevadt Jepsen [1]

http://www.imavs.org/

***Abstract*— This work presents a Detect And Avoid (DAA) concept which was used to enable drone healthcare logistics flight tests adjacent to a Hospital Emergency Medical Services (HEMS) helipad in Denmark. The flights tests were conducted in uncontrolled airspace above a city, where medical helicopters land on average 1-2 times a day.**

The DAA concept utilizes ground-based ADS-B and FLARM receivers to establish situational awareness of the adjacent airspace. The DAA concept together with an associated risk assessment was included in a Specific Operations Risk Assessment (SORA) based application, which received a flight authorization from the Danish Civilian Aviation Authority (CAA).

During a 3 month period, a total of 354 flight tests were conducted totaling 69 hours in air. The operation was inspected by the Danish CAA who have expressed no concerns about the safety of the flight tests.

I. INTRODUCTION

The integration of UAS and crewed aircraft in the lower airspace is by many considered a critical prerequisite to commercial scaling of drone applications [1], [2]. This document presents a Detect And Avoid (DAA) concept for healthcare drone logistics flights between different locations at Odense University Hospital (OUH) in Denmark.

The U-space based aircraft detection is the first step of a Detect And Avoid (DAA) plan to avoid potential mid-air collisions with nearby aircraft. The DAA plan forms part of a Specific Operations Risk Assessment (SORA) based application to be submitted to the Danish Civil Aviation Authorities.

This paper is organized as follows. Section III describes the overall Concept of Operation for the aircraft detection is presented. Section II of the paper gives an overview of the state of the art with the field. In Section IV the DAA plan is described and the associated reliability, robustness, and timing requirements for aircraft detection presented in the SORA guidelines are discussed. Section V describes the technology used for aircraft detection with a focus on reliability. In Section VI the aircraft detection reliability is analyzed using historical data. Finally, section VII concludes on the presented results and discusses future work.

II. STATE OF THE ART

TO BE INSERTED

[1] Kjeld Jensen (kjen@sdu.dk) and Jes Hundevadt Jepsen (jegj@sdu.dk) are with the SDU UAS Center, University of Southern Denmark, Odense, Denmark

[2] Klavs Andersen is with MECK Consult, Smoerum, Denmark. Corresponding email: ka@meckconsult.dk

III. CONCEPT OF OPERATION

During 2021 extensive drone flights will take place at OUH. The purpose is to validate the usability of healthcare logistics drones while at the same time establishing the first U-space airspace in Denmark.

The flights takes place in close vicinity of OUH's Helicopter Emergency Medical Services (HEMS) helipad (EKOH) where medical helicopters land on average 1-2 times a day.

The drone flights are conducted between relevant departments at OUH with a distance of up to a couple of kilometers. At all times will the drone be within Visual Line of Sight (VLOS) to the Pilot In Control (PIC). The flights will take place within 550 m of EKOH which is the distance to the General Practitioners clinic at J.B. Winsløwsvej 9A. The maximum drone height will be 25 m Above Ground Level (AGL).

During flight, the PIC is assisted by a pilot assistant who will support the PIC during the operation including maintaining situational awareness of the nearby airspace. This collaboration is elaborated in the SORA-based application[1] and associated Standard Operating Procedures (SOP). For the purpose of simplicity in this document the term PIC will refer to both the PIC and the pilot assistant. Furthermore, the application describes visual observation as a means to improve the situational awareness of the nearby airspace. For the purpose of simplicity, this document only deals with the U-space based aircraft detection.

A. Detection of aircraft via ADS-B or FLARM

This Concept of Operation (ConOps) assumes that a U-space airspace is established restricting manned aircraft to only enter the U-space airspace, if the aircraft has ADS-B out or FLARM out installed and activated.

Given that the drone operation takes place near EKOH in an urban area where General Aviation (GA) is not permitted to fly below 1000 ft $\approx$ 300 m AGL, essentially only HEMS helicopters approaching the helipad are of concern. The Danish HEMS helicopters are operated by Norsk Luftambulance (NLA). Also, the German helicopters from Nieböl operated by DRF Luftrettung (DRF) occasionally visit EKOH.

Figure 1 lists the ADS-B and FLARM equipment available on HEMS helicopters that have landed at EKOH during the testing period described in section VI. The content of figure

[1] *"Ansøgning om tilladelse til flyvning, mindre end 1 km fra EKOH"*

1 has been confirmed by NLA and DRF in December 2020. It should be noted that LN-OON as the only helicopter is not equipped with ADS-B out. It does have a Mode S transponder installed. LN-OUK is not equipped with FLARM out, however an installation is being planned.

Tail	*ADS-B in*	*ADS-B out*	*FLARM in*	*FLARM out*
LN-OOV	yes	yes	yes	yes
LN-OOZ	yes	yes	yes	yes
LN-OOW	yes	yes	yes	yes
LN-OUK	yes	yes	no	no
LN-OON	no	no	yes	yes
D-HDSY	yes	yes	yes	yes
D-HDSQ	yes	yes	yes	yes

Fig. 1: *ADS-B and FLARM equipment on HEMS helicopters that has landed at EKOH during the testing period described in section VI.*

Figure 2 illustrates the detection of nearby manned aircraft. The ADS-B squitters and FLARM broadcasts are detected by locally placed ground receivers. The received data is transmitted from the ground receivers to a SQL database. The SQL database relays data to a DAA service and also stores data for statistical purposes. The DAA service provides the PIC situational awareness of nearby manned aircraft via a graphical and textual view of the airspace.

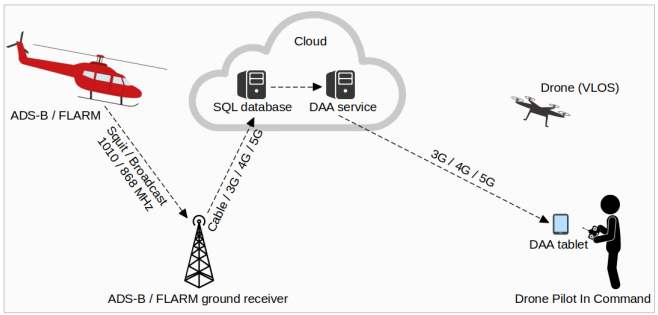


Fig. 2: *DAA architecture for detection of nearby manned aircrafts*

Based on this updated situational awareness the PIC decides appropriate actions if an aircraft approaches. Given that the vast majority of approaching aircraft will be HEMS helicopters landing at EKOH, the only appropriate action will be to immediately land the drone safely.

B. Well Clear volume

The minimum acceptable distance between a drone and a manned aircraft is in this ConOps named the Well Clear (WC) distance. Given that the drone has to yield to all manned aircraft, the PIC has to Remain Well Clear (RWC) i.e. safely land the drone before the WC distance is breached. A contingency or emergency procedure is activated if the WC distance is breached.

The ASTM standard F3442[2] recommends lateral and vertical WC boundaries of 2000 ft ≈ 610 m and 250 ft ≈ 77 m respectively which are used in the following definition of a WC volume. The WC boundary is illustrated in figure 3.

In principle the WC volume follows the drone during flight, however, for simplicity of the operation the WC volume center will be fixed at the EKOH helipad which is close to the center of OUH. The lateral boundary is then extended by the maximum distance from the drone to the EKOH helipad defined in section III. The upward vertical boundary is extended by the maximum height of the drone AGL defined in section III:

$$WC_{lateral} = 610m + 550m = 1160m \quad (1)$$

$$WC_{vertical} = 77m + 25m = 102m\,(AGL) \quad (2)$$

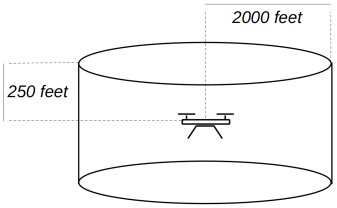


Fig. 3: *Illustration of Well Clear boundaries as suggested by the ASTM F3442F/3442M-20 standard [3]. The lateral boundary is 2000 ft ≈ 610 m. The vertical boundary is 250 ft ≈ 77 m.*

C. Detection volume

When an aircraft approaches the drone operation, the decision whether to continue the current flight or initiate a drone landing must be based on the maximum time duration from detection of the approaching aircraft to the drone that has safely landed.

In this CopOps the decision to land the drone is made by the PIC (vice autonomously), therefore to simplify the decision-making process a detection volume is defined. The size of the detection volume ensures that if a manned aircraft breaches the detection volume, and a safe drone landing is initiated by the PIC, it will be completed before the manned aircraft breaches the WC volume.

In this ConOps the WC volume is cylindrically shaped with its center static at the EKOH helipad as described in section III-B. To keep a simple representation of the detection volume this is similarly cylindrical shaped with the same static center.

Determining the minimum size of the detection volume requires an analysis of latencies and update rates introduced in the DAA plan. This is conducted in section IV which concludes that the estimated maximum time from aircraft detection to the drone has safely landed is 28 seconds. Following this the analysis of historical recorded ADS-B data in section VI was then used to quantify the required minimum size. The proposed size based on the historical analysis is listed below:

[2]Standard Specification for Detect and Avoid System Performance Requirements `https://www.astm.org/Standards/F3442.htm`

http://www.imavs.org/

$$Det_{lateral} = 3000m \quad (3)$$
$$Det_{vertical} = 300m\,(AGL) \quad (4)$$

Figure 4 depicts the WC volume and detection volume centered at EKOH and the proposed sizes listed above.



Fig. 4: *Illustration of the Detection and Well Clear volumes. The illustration is accurate to scale. The vertical lines are caused by the cylinders being represented by a polygon in this illustration. Map content is Copyright by Google.*

D. Restricted airspace

As described in section III-A the ConOps assumes that a U-space airspace is established restricting manned aircraft to only enter the U-space airspace, if the aircraft has ADS-B out or FLARM out installed and activated.

From the principal perspective of the aircraft detection proposed in this document, the U-space airspace must be at least equal to the size of the WC volume defined in section III-B. Any nearby manned aircraft without ADS-B and FLARM are not allowed to enter the U-space airspace (WC volume), and the drone will thus remain well clear of those even if they are not detected inside the Detection volume.

This perspective does, however, not include a buffer for inaccuracies of position and height estimation. Further, it assumes that all nearby manned aircraft operate in compliance with regulations and the restrictions of the U-space airspace.

A mitigating factor to potential manned aircraft not in compliance with regulations or the restrictions of the U-space airspace is the visual observation mentioned in section III. To simplify this visual observation it is proposed that the U-space airspace is defined as larger than the WC volume to enable the observer to easily determine if an aircraft outside but near the WC volume is not in compliance with the U-space airspace restrictions:

- Height: As stated in section III-A the drone operation takes place near EKOH in an urban area where GA is not permitted to fly below 1000 ft $\approx$ 300 m AGL. The probability of having any aircraft without ADS-B or FLARM below 300 m AGL is therefore low and it is proposed that the height of the U-space airspace is set to 300 m AGL similar to the height of the Detection volume.
- Radius: It is proposed to set the lateral radius of the U-space airspace to 3000 m similar to the radius of the Detection volume. This will simplify the visual observation while at the same time minimize the effect of the U-space airspace restrictions imposed on manned aviation flying below 300 m AGL West of Odense city.

IV. DETECT AND AVOID PLAN

The airspace around Odense HEMS is in the SORA defined as ARC-c. The SORA application uses tactical mitigation in the form of a DAA plan. The associated Tactical Mitigation Performance Requirements (TMPR) for ARC-c airspace must be at Medium level. Figure 5 illustrates the DAA plan which is detailed below:

1) **Detect:** Nearby aircraft are detected based on ADS-B squitter and/or FLARM broadcasts. Via a DAA service, the Pilot In Command (PIC)'s establishes an airspace situational awareness using a DAA tablet.
2) **Decide:** Based on the updated situational awareness the PIC decides an appropriate action to either continue the mission or safely land the drone. According to the SORA, the drone must yield to any manned aircraft in the U-space airspace meaning that the drone will have to land if an aircraft approaches. The decision is supported by Standard Operating Procedures (SOP) described in the SORA.
3) **Command:** The PIC commands the drone accordingly. The SORA-based application states that the drone is within VLOS distance to the PIC. The PIC carries a Ground Control Station (GCS) tablet which allows the PIC to update the programmed flight route and other parameters. The PIC also carries a handheld RC transmitter (TX) for direct control of the drone which is used to suspend the programmed flight and land the drone manually if needed.
4) **Execute:** The drone executes the command received via either the TX or the GCS.
5) **Feedback loop:** This loop is continuously updated based on feedback from the drone and updated situational awareness from the DAA tablet.

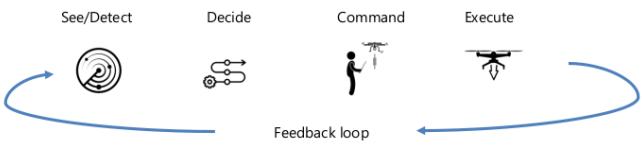


Fig. 5: *Detect And Avoid (DAA) plan for the drone operation.*

A. Tactical Mitigation Performance Requirements (TMPR)

1) Detection reliability and robustness: According to the SORA TMPR requirements listed in figure 6 approximately 90% of all aircraft in the detection volume must be detected.

The SORA D.5.4. TMPR robustness (integrity and assurance) assignment states that: *The allowable loss of function*

http://www.imavs.org/

and performance of the Tactical Mitigation System (TMS) is < 1 per 1000 Flight Hours (10^{-3} Loss/FH). The rate is commensurate with a probable failure condition. These failure conditions are anticipated to occur one or more times during the entire operational life of each aircraft.

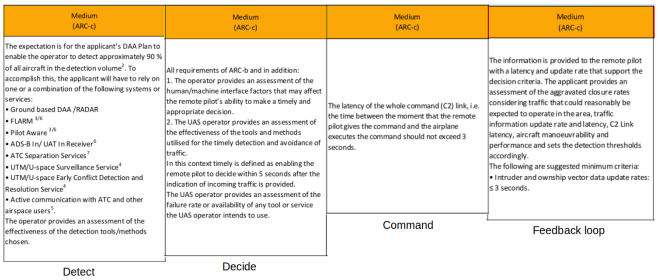


Fig. 6: *TMPR requirements at Medium (ARC-c): This illustration from the SORA documentation lists TMPR requirements at Medium concerning detection reliability, update rate, and latency for "Detect", "Decide", "Command" and "Feedback loop". "Execute" as well as requirements at Low have not been included as they do not elaborate on those requirements.*

2) Latency and update rate: **Detect:** The latency introduced from ADS-B squitters and FLARM broadcasts to presenting this information to the PIC should be a maximum of 5 seconds. The SORA document states that the PIC must look at the display at least every 5 seconds. In total, the detection delay is thus determined to be a maximum of 10 seconds.

Decide: Figure 6 states that the Human-Machine Interface (HMI) must enable the PIC to decide an appropriate action within a maximum of 5 seconds after the information of incoming traffic is available.

Command: Figure 6 states that the C2 link latency should not exceed 3 seconds. The SORA-based application describes a VLOS flight where the PIC carries a handheld RC transmitter for direct control of the drone. RC transmitter update rates are typically around 50 Hz and the latency is significantly lower than 3 seconds. However during manual flight, the Command and Execute latencies are tightly coupled, and the command latency is here kept at maximum to allow some buffer.

Execute: According to the SORA the drone will fly a maximum of 25 m AGL and will be on the ground after a maximum of 10 seconds.

Feedback loop: Figure 6 suggests that aircraft vector data is updated at a rate of a maximum of 3 seconds.

The sum of maximum latencies is used to estimate the maximum time from an aircraft that has squittered or broadcasted its position inside the detection volume to the PIC that has safely landed the drone. This is listed in figure 7.

3) Conclusion: The SORA-based TMPR requirements to the TMS imposes the following requirements to this operation: Approximately 90% of all aircraft in the detection volume must be detected. The DAA detection technology must support an allowable loss of function and performance of less than one instance per 1000 Flight Hours (10^{-3} Loss/FH). The estimated maximum total time from an aircraft that has squittered or broadcasted its position to the PIC has safely landed the drone is 28 seconds. Aircraft vector data should be updated at a rate of a maximum of 3 seconds.

DAA latency	Seconds
Detect:	10
Decide:	5
Command:	3
Execute:	10
Total:	**28**

Fig. 7: *Estimation of the maximum time from an aircraft has squittered or broadcasted its position to the PIC has safely landed the drone.*

V. DAA DETECTION TECHNOLOGY

In this section the technology used for detection of nearby manned aircraft is presented, and compliance with the TMPR requirements outlined in section IV-A.3 is discussed. The section is divided into an elaboration of TMPR requirements with regards to the DAA detection technology followed by a description and assessment of the principal components of the DAA architecture illustrated in figure 2.

A. Allowable loss of function and performance

According to the TMPR requirements listed in section IV-A.3, the allowable loss of function and performance of the TMS is 10^{-3} Loss/FH.

In this operation, the TMS is the DAA technology architecture illustrated in figure 2 and the principal components thereof. The sum of probabilities of loss of function and performance for each component including their respective communication links must thus meet the below requirement:

$$P_{Loss:\,Receiver} + P_{Loss:\,SQL} + P_{Loss:\,DAA} + P_{Loss:\,Tablet} < 10^{-3} Loss/FH \quad (5)$$

The assessment of these probabilities are discussed in the below descriptions of the components using both quantitative and qualitative arguments. To simplify the argumentation, where applicable, the theoretical assumption is used that if continuous flights were conducted, then 10^{-3} Loss/FH approximates an operation of 42 days with a maximum of 1 loss of function or performance.

B. ADS-B ground receiver

The ADS-B receiver location is 900 m from EKOH. As illustrated in figure 4 the Detection volume consists of mainly urban areas. The terrain is flat with a topographical relief of only a few meters. The antenna is installed at a height of approximately 8 m AGL. While this does not provide a clear line of sight to low-altitude aircraft in the entire detection volume, the tests in section VI documents that the reception capabilities by this setup is sufficient.

http://www.imavs.org/

http://www.imavs.org/

The ADS-B ground receiver hardware components: a radio receiver and an embedded computer are standard commercial grade. The power supply has a redundant battery backup supporting continued operation for at least 30 minutes during a power outage. There is no available documentation of reliability parameters such as Mean Time Between Failure (MTBF), however, the hardware has proven to be very reliable[3] and has not had any downtime during the aircraft detection reliability assessment period described in section VI-B.

The ADS-B ground receiver internet connection to the cloud-based SQL database described in section V-D has in Feb. 2021 been upgraded to facilitate a redundant fallback to a 4G modem connection.

The ADS-B ground receiver software is developed and maintained by SDU. It is in part based on open source software which has been improved and extended over the past years. During the period of recording historical data which is analyzed in section VI, the software exhibited an error that caused the detection to sometimes stop functioning after loss of the internet connection. The error has not occurred since the introduction of the redundant internet connection and software changes to fix the error.

The operational state and functionality of the ADS-B receiver is continuously monitored using an independent monitoring service. This service monitors the full functionality of the ADS-B receiver. If the latest aircraft squitter or broadcast sent to the cloud-based SQL database is more than 60 seconds old, an alert is sent via SMS to the PIC.

Based on the above description of the ADS-B ground receiver reliability and redundancy and on the results from the reliability assessment in section VI it is concluded that the detection rate using this receiver alone is > 90% and that $P_{Loss:\,Receiver} << 10^{-3} Loss/FH$.

C. FLARM ground receiver

Aircrafts emitting FLARM are detected by two independent FLARM receivers:

- An Open Glider Network (OGN) based FLARM receiver is located at HCA Airport (EKOD) approximately 10 km from EKOH. The antenna is installed at a height of approximately 14 m AGL providing a clear line of sight in the lateral direction. This receiver provides OGN-based FLARM detections to the SQL database. This setup provides detailed data of FLARM-equipped aircraft operating around EKOH.
  The receiver has only been in operation for a short period of time though and it does not have a redundant power source or internet connection. It is therefore not considered a reliable source of FLARM detections.
- A PowerFLARM-based receiver is located 900 m from EKOH. The antenna is installed at a height of approximately 8 m AGL. This does not provide a clear line of sight to low-altitude aircraft in the entire detection volume. This is a recent installation and the FLARM detection capabilities are yet unknown. This receiver is thus not considered a reliable source of FLARM detections.

Overall the FLARM detection appears to be working well but is at this point not considered reliable due to pending reliability upgrades and the lack of statistical data to document reliability.

D. Cloud-based SQL database

The SQL database runs on a virtual server hosted by HostEurope[4], a server hosting company with a Service Level Agreement (SLA) that guarantees a minimum 99.95% uptime based on monthly mean. The host testing company HOSTtest.de has measured 99.88% uptime. HostEurope has been used for hosting the database for more than 5 years.

The installed SQL server instance with no real-time redundancy is likely the weakest point of the SQL database, however, it has worked for years without any unscheduled downtime or other problems and did not contribute to detection loss during the reliability assessment conducted in section VI.

The operational state of the SQL database is continuously monitored using an independent monitoring service. This service performs a database request every minute which validates that the database has been updated with new data from either of the ground receivers. If this validation fails it is assumed that the database is not operational and an alert is sent via SMS to the PIC.

The measured uptime of the virtual server (99.88%) corresponds to a downtime of 10.5 hours per year which from a purely statistical perspective contradicts the requirement of 42 days continuous service stated in section V-A. It should be noted though that the vast majority of downtime is scheduled maintenance work by the hosting company which takes place during night time and is announced to customers days in advance. Based on this, the extra integrity provided by the independent monitoring service and the results from the reliability assessment in section VI it is concluded that: $P_{Loss:\,SQL} << 10^{-3} Loss/FH$.

E. Cloud based DAA service

The Cloud based DAA service for this operation consists of a web page retrieving data from the SQL database via a Restful Web Service at a rate $\geq$ 1 Hz. The web page shows a map of the detected aircraft in the vicinity of EKOH. Below the map is a list of the aircraft including those not broadcasting their position and thus not shown on the map. The page may show additional DAA guidance on the map or above or below the list of aircraft.

The DAA service is hosted on a webserver running on the same virtual server as the SQL database. Like the SQL server instance, the webserver has no real-time redundancy but has worked for years without any unscheduled downtime. Should any downtime occur without warning, the PIC will become

[3]Measured February 22nd, 2021 the current ADS-B ground receiver computer has a registered uptime of 991 consecutive days without the need for a restart.

[4]https://hosteurope.com

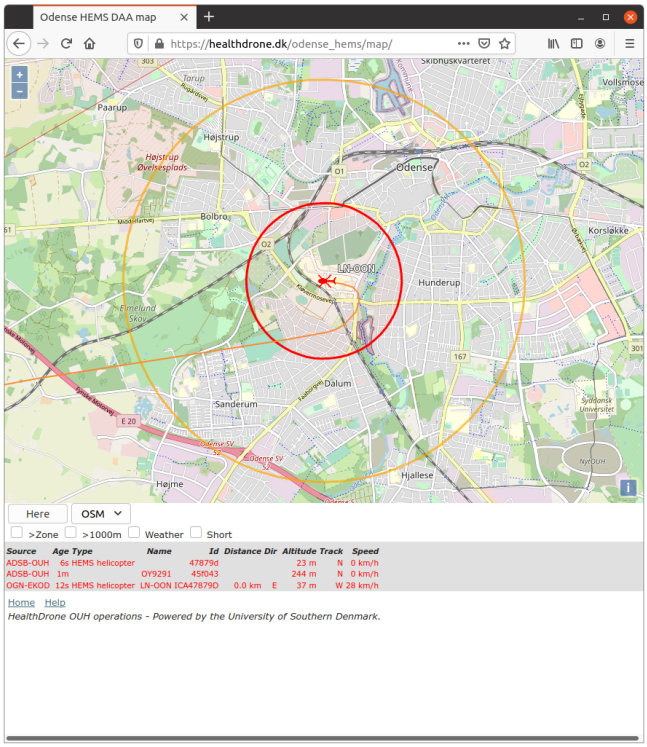


Fig. 8: *The map view is available via the cloud-based DAA service. The current view shows how the helicopter LN-OON (which as documented in figure 1 is the only HEMS helicopter not equipped with ADS-B) is listed as both a Mode S aircraft without position and is listed and shown on the map based on FLARM data. OY9291 is another aircraft in the vicinity that has only Mode S. The aircraft are all marked red because they report an altitude less than the height of the Detection Volume (300 m).*

aware just like if the DAA tablet looses connection to the internet (see section V-F). Based on this it is concluded that: $P_{Loss:\,DAA} << 10^{-3} Loss/FH$.

F. DAA tablet

A tablet based on Android or IOS will be used to provide the PIC the situational awareness required for DAA. The tablet is considered the overall weakest component with regards to the 10^{-3} Loss/FH requirement. Considering a theoretical continuous operation for 42 days there is a risk that either the wireless internet connection or the tablet itself or the web browser running on the tablet experiences temporary problems. While this is not expected to happen often, the probability of loss of function or performance is likely > 10^{-3} Loss/FH. This will be mitigated by SOP and independent monitoring of the operational state:

Operational procedures defined in SOP's will specify requirements for using a dedicated tablet, proper charging before and during operation, restart and testing of the tablet before use, as well as a readily available backup tablet or phone running on a different cellular network. Also, the procedures will specify that the PIC continuously monitors if the web page is "frozen" indicated by error messages or lack of updated data.

The operational state of the web page running on the tablet is continuously monitored using an independent monitoring service. This service performs a request to the DAA service each minute which validates that the web page is continuously requesting updates from the DAA service. If this validation fails it is concluded that the DAA tablet is not operational and an alert is sent via SMS to the PIC. The SMS transmission time has experimentally been measured to be < 15 seconds which means that the PIC will be alerted < 75 seconds after a loss of function.

Based on the operational procedures to mitigate the risk of a DAA tablet fault and the extra integrity provided by the independent monitoring service it is concluded that: $P_{Loss:\,Tablet} << 10^{-3} Loss/FH$ under the condition that a probability of loss of function or performance where the PIC is alerted > 10^{-3}/FH is acceptable. This is discussed further in section VII.

VI. AIRCRAFT DETECTION RELIABILITY ASSESSMENT

In this section the reliability of the aircraft detection described in section III-A and V is assessed based on an analysis of historical recorded data from the ADS-B ground receiver. The purpose is to substantiate the arguments presented in section V concerning the reliability of the DAA architecture.

The recorded data is compared to a hand-written log of all EKOH HEMS activities. This hand-written log is maintained by the nurse at the OUH emergency room who is in direct contact with the HEMS helicopters via the SINE radio network. The following sections describe the method used and the results.

A. Method

The recorded data is continuously saved in the SQL database. Each time an ADS-B squit is received from a ground receiver, a record is stored in the database[5]. The record contains among other information the time of reception by the ground receiver, the ADS-B ICAO ID, the geographical position, and altitude if available. The following procedure was used to process the data:

1) To exclude data of no interest a search was performed for ADS-B squitter records with altitude $\leq$ 1000 m. Since there is only data from one ADS-B ground receiver a lateral limitation was not necessary.
2) To establish a list of aircraft all found ADS-B squitter records containing a valid geographical position traversed. Each was tested if the geographical position (including altitude if available) was within the detection volume defined in section III-C using a radius of 3000 m and a height of 300 m AGL. If so the aircraft ICAO ID was added to the list of aircraft if not already listed.

[5]FLARM broadcasts are stored similarly to ADS-B data, but FLARM data is not included in the analysis as the FLARM ground receiver has only been connected to the SQL database since December 2020.

http://www.imavs.org/

http://www.imavs.org/

3) The NLA helicopter LN-OON listed in table 1 was manually added to the list of aircraft. It contains only a Mode S transponder so no positions are stored in the records.
4) For each aircraft in the aircraft list a search was made in the found ADS-B squitter records for records squittered by that aircraft. This subset was then sorted by the time of reception and traversed to identify individual flights. A period of 300 seconds without a squit triggers a new flight. Fligths with less than 15 seconds between the first and last squit were disregarded as stray receptions due to low reception quality, most likely from aircraft at too low altitudes.
5) Each flight was analysed further: If a recorded flight began outside and ended inside the detection cylinder it was marked "Arrival". Similarly, if a recorded flight began inside and ended outside the detection cylinder it was marked "Take-off". All other flights remaining outside or without any geographical positions were marked inconclusive by a "?".
6) For each flight marked "Arrival" the first squit after entering the detection volume and WC volume respectively was identified. Based on this the time inside the detection volume was calculated as $(time_{first\ squitin\ det.\ vol.} - time_{first\ squit\ in\ WC\ vol.})$. Similarly, the time before reaching the detection volume was calculated as $(time_{first\ squit\ with\ position} - time_{first\ squitin\ det.\ vol.})$.
7) All flights were sorted according to the timestamp of the first squit and the resulting flight log was formatted for comparison. The hand-written log was manually entered into a spreadsheet. A manual comparison of the two logs was conducted identifying all matching flights defined as a clear relation between the landing time and take-off time in the flight log and the corresponding entry in the spreadsheet.

B. Results

The period of comparison is June 11th, 2020 to December 1st, 2020 decided by available comparable log material. This amounts to 179 days in total.

A total of 295 arrivals at EHOH were registered by comparing the ADS-B based flight log with the handwritten log. The average number of arrivals per day is thus 1.6.

The ADS-B ground receiver received and stored in the SQL database 290 of the 295 arrivals. The statistical detection is thus 98.3%.

Figure 9 shows a distribution of Odense HEMS arrivals with ADS-B based on the time from the first position received from the aircraft to the aircraft breaching the Well Clear volume.

Figure 12 shows the number of arrivals across the time of day. While there are flights at all times, the majority appear to be between 8 and 22 hours. As seen in figure 13 the arrivals do not exhibit any apparent pattern with regard to the days of the week.

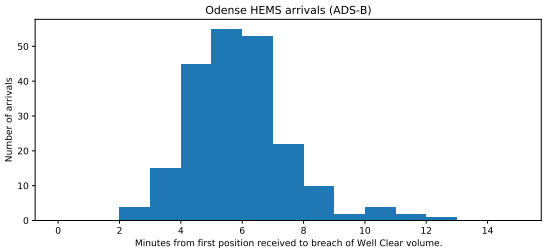


Fig. 9: *Distribution of Odense HEMS arrivals with ADS-B based on the time interval from the first position received from the aircraft to the aircraft breaching the Well Clear volume.*

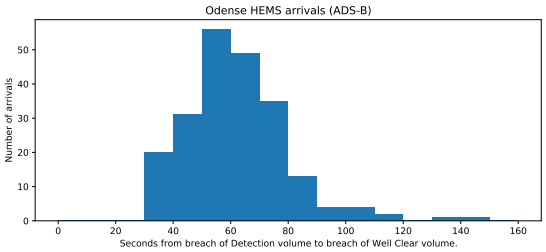


Fig. 10: *Distribution of Odense HEMS arrivals with ADS-B based on the time interval from the aircraft breaching the Detection volume to the aircraft breaching the Well Clear volume.*

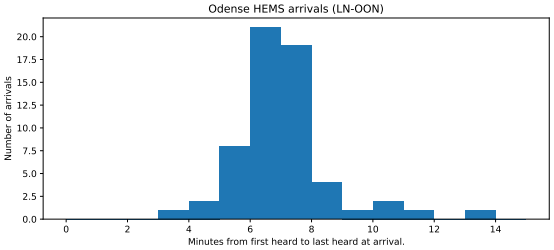


Fig. 11: .

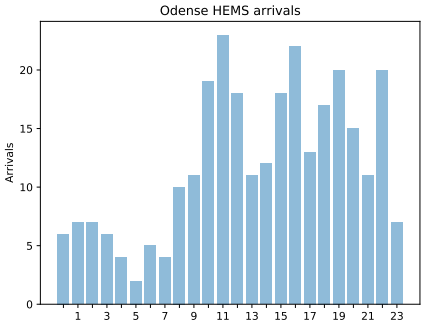


Fig. 12: *Distribution of Odense HEMS arrivals across the hours of the day.*

The list of aircraft and flights from the processed ADS-B squitter records is available for review at the link below. Clicking the individual flights presents the flight squitters on

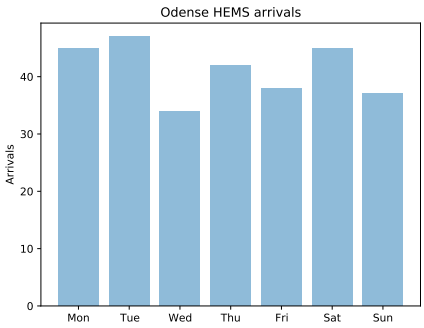


Fig. 13: *Distribution of Odense HEMS arrivals across the days of the week.*

a map (if positions were squittered) and an altitude plot (if altitude was squittered) as seen in figure 14. Zooming into the map enables clicking each ADS-B squit to view relevant information. Figure 1 lists the different HEMS helicopters that landed at EKOH during the period of the historically recorded data.

`https://uaswork.org/project/healthdrone/odense_hems_sora`

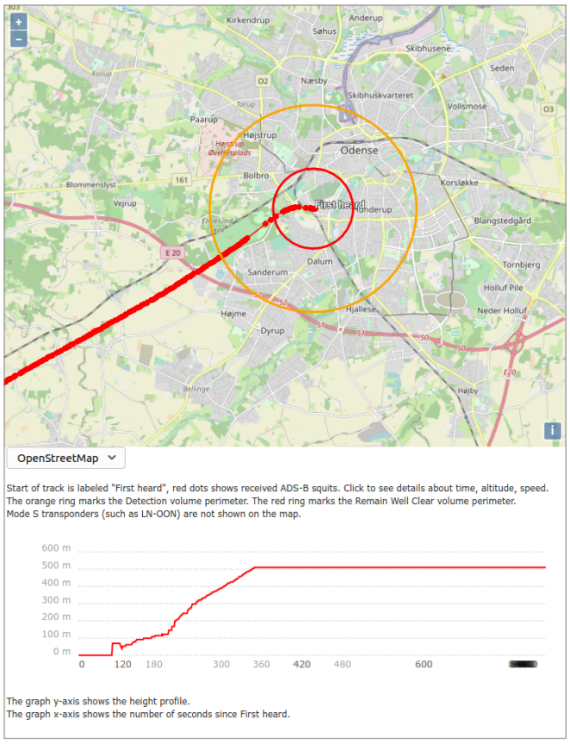


Fig. 14: *Map showing a flight based on the processed ADS-B squitter records.*

C. Discussion

As described above 290 out of 295 arrivals were successfully retrieved and stored in the SQL database. Of the remaining 5: at least 4, and very likely all 5, were not detected due to a brief internet outage triggering the known ADS-B Ground receiver software error described in section V-B. This leads to the conclusion that while the statistical detection based on approximately 6 months of data is 98.3%, the future detection rate will likely be higher.

VII. CONCLUSION

The concept for aircraft detection proposed in this document defines two airspace volumes shaped as cylinders around EKOH. The innermost Well Clear volume based on the ASTM standard F3442 defines the airspace where, if breached by a manned aircraft, the drone must have landed. The outermost Detection volume defines the airspace where, if breached by a manned aircraft in compliance with the U-space restrictions, the drone pilot must initiate a safe landing.

Aircraft detection is performed using a DAA architecture consisting of ADS-B and FLARM ground receivers that send detections to a SQL database. A DAA service reads the SQL database and presents a DAA view in a web browser on a tablet observed by PIC.

While the FLARM receivers provide valuable supplementary information, they are not considered a critical part of the DAA architecture and are thus not included in the TMPR compliance assessment.

The reliability of aircraft detection in the Detection volume was analysed by comparing historical ADS-B data recorded near EKOH during 6 months in 2020 to a log maintained by OUH. The analysis documents that 290 of 295 registered helicopter arrivals (98.3%) were successfully detected. 4 of the remaining 5 (1.4%) were not detected due to a known software error.

Only one HEMS helicopter (LN-OON) operates at EKOH without ADS-B. During the aircraft detection reliability assessment period, LN-OON was consistently detected via the ADS-B ground receiver using the Mode S reply.

The proposed concept for aircraft detection is compliant with all TMPR requirements on timing and latency. The historical ADS-B data documents that the aircraft detection reliability is on the order of 10 times better than the TMPR requirement of approximately 90%. TMPR specify a maximum loss of function and performance to 10^{-3} loss/FH. For the proposed concept of aircraft detection, this is a very challenging requirement to meet without full redundancy on all principal components, especially when a tablet with wireless communication is used to present DAA information to the PIC. Operational procedures and independent monitoring of system integrity are introduced to mitigate the risk and consequence of exceeding this requirement. These mitigations will not improve reliability per se but will ensure that a contingency is triggered and thus a safe landing is conducted. It should be noted that while the 10^{-3} loss/FH requirement is more challenging to comply with than the detection reliability for the proposed concept of aircraft detection, an undetected aircraft approaching EKOH likely poses a higher risk than a brief loss of detection functionality at the order of maximum 75 seconds.

http://www.imavs.org/

The historical ADS-B data was used to define a distance from the Detection volume to the Well Clear volume which ensures adequate time for a safe landing. The Detect And Avoid plan documents that a maximum of 28 seconds are needed for detecting an aircraft and landing the drone. The historical ADS-B data documents that given the proposed size of the volumes 7% of the arrivals completed this distance in 30-40 seconds, 11% in 40-50 seconds and the remaining arrivals took longer time. It should be noted that all detected aircraft were detected at least 120 seconds before entering the Well Clear volume.

Based on the above it is concluded that the proposed concept for aircraft detection at EKOH is compliant with requirements defined in the EU regulation for drone operations in the specific category. This conclusion is based exclusively on the U-space based detection. It does thus not take into account the visual observation described in the concept of operation which adds an additional layer of safety.

ACKNOWLEDGMENT

This work is part of the HealthDrone project [4] funded by the Innovation Fund Denmark [5]. The authors would like to thank Mathias Vinter (Holo), Hans-Christian Rasmussen (Falck A/S), and the Odense University Hospital (OUH) Emergency Department and Facility Management for their contributions.

REFERENCES

[1] S. Michaelides-Mateou, *Challenges and Trends in the Aviation Industry: Integrating UAVs in Non-segregated Airspace*. Cham: Springer International Publishing, 2023, pp. 377–409. [Online]. Available: https://doi.org/10.1007/978-3-031-32037-8_13

[2] M. Grote, A. Pilko, J. Scanlan, T. Cherrett, J. Dickinson, A. Smith, A. Oakey, and G. Marsden, "Sharing airspace with uncrewed aerial vehicles (uavs): Views of the general aviation (ga) community," *Journal of Air Transport Management*, vol. 102, p. 102218, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0969699722000394

[3] "F3442/F3442M-20 Standard Specification for Detect and Avoid System Performance Requirements," ASTM," Standard, July 2020.

[4] HealthDrone. (2024, Feb.) HealthDrone project. [Online]. Available: https://sundhedsdroner.dk/index.php?page=the-project

[5] Innovation Fund Denmark. (2024, Feb.). [Online]. Available: https://innovationsfonden.dk/en

http://www.imavs.org/

Extended Reception Coverage and Spoofing Localization of Direct Remote ID utilizing Ad-Hoc Mesh Networking

M. F. Høffer [1], V. D. Herlev[1], G. Maalouf[1], J. H. Jepsen[1], E. Winkler[1], I. Panov[2], and K. Jensen[1]
[1]SDU UAS Center, University of Southern Denmark, Odense
[2]Department of Business Development and Technology, Aarhus University, Aarhus

http://www.imavs.org/

ABSTRACT

Direct Remote Identification (DRI) identifies and localizes nearby Unmanned Aerial Vehicles (UAVs) by transmitting the location and identity of the UAV and operator via Bluetooth or Wi-Fi. As of 1 January 2024, the EU mandates remote identification for UAVs in category C. Current DRI systems have a limited range and lack digital signatures, making them vulnerable to spoofing.

This paper addresses the first limitation by using a Flying Ad-Hoc Network (FANET) to extend the range of DRI messages with a meshing algorithm. The second limitation is tackled by investigating RSSI localization methods. Spoofing detection is achieved by comparing estimated and broadcast locations. The meshed system was tested and reliably received messages at a 1.6 km range, previously unattainable. Localization experiments demonstrated the approximate localization of both static and moving DRI devices using a receiver network.

1 INTRODUCTION

As of 1 January 2024, the European Union's 2019/947 regulation [1] mandates that Unmanned Aerial Vehicles (UAVs) in category C use a modern remote identification system. Two modes exist: Network Remote Identification (NRI) is used within U-space airspace, while Direct Remote Identification (DRI) is used outside of U-space.

Both modes transmit operator and UAV information, but NRI broadcasts globally via the internet, while DRI broadcasts locally. Only DRI is required for operations outside U-space, allowing authorities and the public to receive information within a certain range through software applications.

The DRI standard [2] supports four wireless technologies: Bluetooth Legacy Advertising, Bluetooth Long Range, Wi-Fi Beacon, and Wi-Fi Neighbor Aware Network (NAN). These technologies have a limited range of a few hundred meters [3].

Moreover, the DRI standard lacks specifications for verifying message through digital signature. While the Internet Engineering Task Force is working on DRI security [4, 5], current specifications do not address this, making spoofing a significant threat to UAV situational awareness and airspace security. A malicious actor could transmit false DRI messages, evading detection or simulating one or more UAVs with a single transmitter.

The two weaknesses of the current DRI standard are illustrated in Fig. 1. The contribution of this work is to 1) demonstrate how the range of DRI can be increased with the use of mesh networking to create a Flying Ad-Hoc Network (FANET), and 2) investigate how a device transmitting fake DRI messages can be localized using a receiver network.

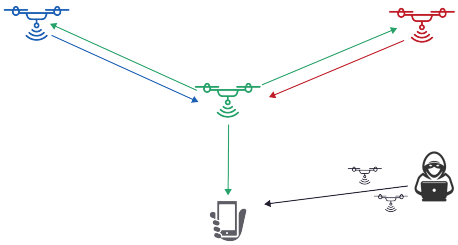


Figure 1: The green UAV is able to receive DRI messages from the blue and red UAV. However, the DRI scanner application only receives DRI messages from the green UAV. The blue and red UAV have no knowledge of each other. The system is vulnerable to spoofing devices.

The paper is organized as follows: Section 2 of the paper gives an overview of the state of the art with the field of ad-hoc mesh networks for UAVs and localization using wireless sensor network. Section 3 describes the architecture of the presented DRI FANET, including 1) how it extends the range of DRI messages (Section 3.1) and 2) how it localizes DRI spoofing devices (Section 3.2). The results are discussed in Section 4. Finally, Section 5 covers the conclusion and future work.

2 STATE OF THE ART

Ho et al. argues that flooding is a viable broadcasting algorithm for high reliability in very dynamic networks [6]. The argument is based on the fact that keeping accurate information of neighboring nodes is difficult for high-mobility nodes. Simulations are performed that visualize how the flooding algorithm performs in terms of the number of collisions, duplicate messages, and packet loss as a function of the node speed.

In [7], the difficulties related to broadcasting information in a mobile ad-hoc network are addressed. These concerns can be directly transferred to FANETs. The paper focuses on the broadcast storm problem, which is a collection of problems associated with flooding. To combat issues regarding

http://www.imavs.org/

redundancy, network congestion and packet collisions, the authors propose different broadcast algorithms such as probabilistic and location-based schemes, and analyze their performance.

In [8], the authors propose a solution to reduce broadcast redundancy in wireless mobile networks. The proposed solution is the Scalable Broadcast algorithm (SBA), which uses 2-hop neighbor information to make local decisions with very little overhead. Other neighbor knowledge methods exist, where the routing decision is not made locally but instead predetermined by the previous transmitter [9, 10].

Williams and Camp categorize and use simulation to compare different broadcasting algorithms usable in mobile ad-hoc networks [11]. The categories include simple flooding, probabilistic methods, area-based methods, and neighbor knowledge methods. In terms of redundant retransmissions of packets in dense networks, their results show that simple flooding performs the worst, while the neighbor knowledge methods perform the best. Additionally, they show that neighbor knowledge methods that do not make local decisions perform the worst in high mobility networks, in terms of packet delivery ratio.

Localization of Wireless Sensor Nodes (WSNs) is an active area of research as more wireless devices with a need for a location estimate enter the market [12]. A common approach in localization is to use anchor nodes that broadcast their known locations to estimate the position of other nodes [13]. In the context of DRI, the roles are instead reversed, as the goal is to localize a spoofing device based on the information the device itself transmits. Localization of WSNs can broadly be categorized in two categories; range-based and range-free localization [14]. Range-based localization methods utilize a distance or angle measurement between an unknown position and several anchor nodes with known positions.

Estimating the distance to anchor nodes is often done using the log-distance path loss model given that the received power, path loss exponent, and a reference received power at e.g $1\mathrm{m}$ is known[15, 16]. The main challenge in using the log-distance path loss model lies in estimating the path loss exponent, which is dependent on the environment. In [16] the path-loss exponent and reference received power are numerically estimated through a least squares minimization.

Blumental et.al proposes a range-based localization method called Weighted Centroid Localization (WCL) algorithm in [17]. Sensor nodes with unknown positions can estimate their position as a weighted sum of the anchor nodes' positions, where the weight is determined based on the distance between the sensor node and anchor nodes. In [18] a range-free extension of the WCL algorithm is proposed, the Weight Compensated Weighted Centroid Localization (WCWCL) algorithm, where the weights no longer rely on a distance measurement, but purely the signal strength. Simulation and experimental results show an improved localization accuracy compared to WCL.

3 DRI FLYING AD-HOC NETWORK (FANET)

The conceptional idea of presented DRI FANET is illustrated on Fig. 2. Each DRI device in the FANET is based on an XIAO ESP32-C3 development board [19], which is a suitable candidate for an embedded platform used in mesh network applications on UAVs due to the following reasons; it has both Bluetooth and Wi-Fi capabilities; it has an onboard battery circuit; it has a low price; the weight is $10\,\mathrm{g}$, and; it has the size of $21\,\mathrm{mm} \times 17.5\,\mathrm{mm}$. For further details, we refer to the Master's Thesis of M. F. Høffer and V. D. Herlev [20].

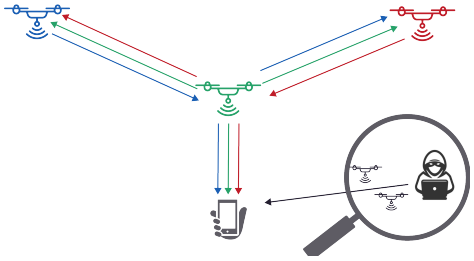


Figure 2: All UAVs are aware of each other and the DRI scanner application receives DRI messages from all UAVs. This is achieved by the green UAV forwarding messages from the others. Additionally, the spoofing devices are localized.

3.1 *Extending the coverage range of DRI*

The conceptual idea of extending the coverage range of the DRI is illustrated on Fig. 3, and is obtained by having individual nodes receive the DRI messages and retransmit them based on a forwarding rule. Since only broadcasting is considered, this rule will be determined by a broadcast algorithm. If the forwarding rule was to retransmit all packets blindly, the packets would be broadcast. However, the network would quickly become congested because of the packets moving in loops.

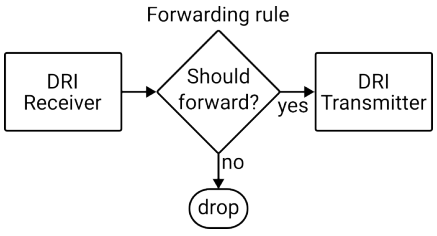


Figure 3: DRI ad-hoc mesh network principle. A DRI receiver listens for DRI messages, if the forwarding rule decides that the message should be forwarded, it is rebroadcast using the DRI transmitter, otherwise, it is dropped.

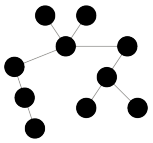


Figure 4: Sparse network where the structure can be described as a tree or a graph without cycles.

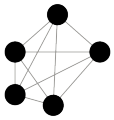


Figure 5: Dense fully connected network. Every node is connected to every other node.

3.1.1 Comparison of different broadcast algorithms

Table 1 summarizes the different broadcast algorithms that have been considered for the DRI FANET; In **simple flooding**, every node forwards every packet exactly once; The three probabilistic methods **counter-based**, **distance-based**, and **location-based** use a threshold value to determine whether to forward a packet; The two neighbor-based approaches, Scalable Broadcast Algorithm (SBA) and Ad-Hoc Broadcast Protocol (AHBP), use two-hop neighbor knowledge to determine whether to forward a packet, where: 1) In **SBA**, if a node has neighbors that are not covered already, it will rebroadcast the packet, and 2) In **AHBP** every node selects a list of Broadcast Relay Gateways (BRGs), which are the only nodes allowed to forward the packet. The BRGs are selected such that all nodes in the network will be covered by a broadcast. Some of the algorithms require attaching additional information to the DRI messages. Modification of DRI messages can result in standard receivers not being able to receive the forwarded message, which is not desirable.

In [11], Williams and Camp categorize and compare different broadcast algorithms in simulation. Their results show that the two probabilistic methods counter-based and location-based perform worst in dense networks (Fig. 5), while AHBP performs worse for networks with high mobility. Additionally, the probabilistic methods do not guarantee delivery in certain network topologies.

It is assumed that networks consisting of UAVs are sparse (Fig. 4) and have a highly dynamic topology, as the UAVs are expected to move. In this case, both SBA and simple flooding are applicable algorithms. However, SBA has increased overhead in keeping track of neighboring nodes. Additionally, SBA does not rebroadcast messages at the ends of branches and therefore has reduced coverage compared to simple flooding. For this reason, the simple flooding algorithm is chosen to extend the range of DRI.

3.1.2 Implementation

The simple flooding algorithm is implemented on the ESP32-C3 as illustrated on Fig. 6. Additionally, integration with a flight controller is achieved through MAVLink[21]. This makes the UAV aware of other UAVs on the network, as well as making other UAVs on the network aware of the UAV itself.

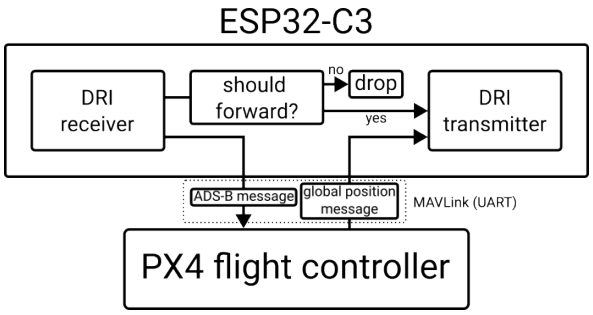


Figure 6: Integration of the DRI with a PX4-based flight controller for extending its coverage range using the simple flooding broadcast algorithm.

3.1.3 Results

The experimental validation of extending the DRI range is visualized on Fig. 7. The DRI FANET consists of eleven ESP32-C3s, each with the simple flooding broadcast mesh algorithm implemented.

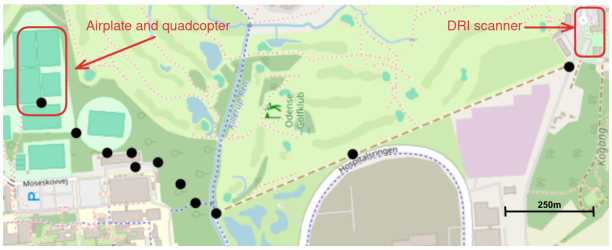


Figure 7: Location of relay nodes used for broadcast mesh network verification. AirPlate[22] and quadcopter are placed on one end of the chain and a DRI scanner on the other end. DRI messages transmitted by the Airplate and quadcopter will be forwarded through the chain of relay nodes using the developed broadcast algorithm.

Two PX4-based [23] flight controllers are used in the setup; One of the flight controllers is used for controlling a multirotor UAV that will be flying and transmitting DRI messages at 10Hz, and; One flight controller is connected to a Ground Control Station (GCS), used as a DRI scanner through QGroundControl[24]. Both flight controllers are communicating with an ESP32-C3 using the MAVLink protocol, as illustrated on Fig. 6. An AirPlate [22], a spinout from the GENIUS project [25] is used to verify that the FANET also supports devices without mesh network capabilities. The airline distance between the two ends of the relay chain is approximately 1.6 km. Fig. 8 shows the frequencies at which the messages from the quadcopter and AirPlate arrive at the DRI scanner. The altitude of the quadcopter is

http://www.imavs.org/

Algorithm	Can be implemented without modifying DRI?	Most applicable for sparse or dense network?	Handles networks with dynamic topology well?
Simple Flooding	Yes	Sparse	Yes
Counter-Based	Yes	Dense	Yes
Distance-Based	Yes	-	-
Location-Based	No	Dense	Yes
SBA	Yes	All	Yes
AHBP	No	All	No

Table 1: Comparison of different broadcast algorithms, where the properties relevant in the context of a DRI FANET are shown.

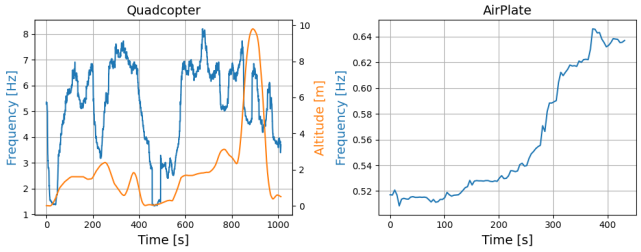


Figure 8: Moving average, with a window size of 150, of the frequency at which DRI messages arrive at the DRI scanner from the quadcopter and AirPlate respectively. The quadcopter broadcasts at 10 Hz, while the AirPlate broadcasts at 1 Hz.

plotted alongside the frequency, to show that the frequency is lowest when the quadcopter is on the ground.

The message frequency can be used as a measure for the delivery ratio, as it ideally should be 10 Hz and 1 Hz for the quadcopter and AirPlate respectively. To validate the implemented solution, it was verified that all of the messages arriving at the DRI scanner came from the last node in the relay chain exclusively. A video of the experiment can be found in [26].

3.2 Localization of a DRI spoofing device using FANET

As mentioned, the DRI specification does not implement any sort of verification of DRI messages. As a result of this, all of the information in the DRI messages could be spoofed. Figure 9 illustrates the potential issues that can arise from this vulnerability. This section investigates if it is possible to localize a device transmitting DRI messages. This can then be used to determine if a device is spoofing its position. Detecting if other information in the DRI messages is incorrect, such as the UAV and operator ID, is not considered.

One measurable metric that can be used for localization is the Received Signal Strength Indicator (RSSI). In the presented work, two localization methods based on RSSI measurements are evaluated; 1) Multilateration, a range-based RSSI localization method, and 2) Weight Compensated Weighted Centroid Localization (WCWCL), a range-free RSSI localization method.



Figure 9: Example of a single device spoofing HCA airport. Note that the experiment is performed in an isolated environment and therefore caused no impact on real-world air traffic.

3.2.1 Multilateration

Based on the log-distance path loss model, the distance between transmitter and receiver can be estimated using Eq. (1), where RSSI(d_0) is a reference RSSI measurement made at a distance d_0 and n is the path loss exponent.

$$d = d_0 10^{\frac{\text{RSSI}(d_0)-\text{RSSI}(d)}{10n}} \tag{1}$$

In the context of DRI spoofing, there will be one transmitter and N receiver nodes. For each receiver node n_i with coordinates (x_i, y_i), a corresponding distance estimate d_i to the transmitter will be calculated. Each distance estimate will correspond to a circle with center (x_i, y_i) and radius d_i, that the transmitter can be located on. In an ideal case, the circles intersect at exactly one point (x, y) which can be determined analytically. Due to inaccuracies in the distance estimates this is however unlikely. If several anchor nodes $n > 3$ are used, an over-determined system of equations can be formulated as seen in Eq. (2).

$$\begin{aligned} (x-x_1)^2+(y-y_1)^2 &= d_1^2 \\ &\vdots \\ (x-x_n)^2+(y-y_n)^2 &= d_n^2 \end{aligned} \tag{2}$$

http://www.imavs.org/

http://www.imavs.org/

The system of equations can be represented on matrix form as $\boldsymbol{Ax} = \boldsymbol{b}$, which can be solved using Eq. (3), where $\hat{\boldsymbol{x}} = (\hat{x}, \hat{y})$ is the estimated transmitter position.

$$\hat{\boldsymbol{x}} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b} \tag{3}$$

3.2.2 WCWCL

Range-free localization does not rely on a distance measure, and therefore not on an estimate of the path loss exponent. In [18] a novel range-free localization algorithm, called Weight Compensated Weighted Centroid Localization (WCWCL), is presented. In WCWCL each anchor node in the receiver network is assigned a weight, which is calculated based on the RSSI value as shown in Eq. (4), where g is the degree that determines the contribution from each receiver node.

$$W_i = \frac{w_i}{\sum_{j=1}^{n} w_j} = \frac{\sqrt{\left(10^{\frac{RSSI_i}{10}}\right)^g}}{\sum_{j=1}^{n}\sqrt{\left(10^{\frac{RSSI_j}{10}}\right)^g}} \tag{4}$$

A weight compensated weight Wc_i is then calculated to increase the weight of receiver nodes that are closer to the transmitter, this is shown in Eq. (5), where n is the number of receiver nodes.

$$Wc_i = \frac{W_i \cdot n^{2W_i}}{\sum_{j=1}^{n}(W_j \cdot n^{2\cdot W_j})} \tag{5}$$

Based on the assigned weights the position of the transmitter P is then calculated as the sum of the product of the weights Wc_i and the position of each receiver node P_i, as shown in Eq. (6).

$$P = \sum_{j=1}^{n}(Wc_i \cdot P_i) \tag{6}$$



Figure 10: Receiver node with an ESP32-C3 and a 2.15 dBi stacked dipole whip antenna mounted on a bamboo stick, used in the localization experiment.

3.3 Results

3.3.1 Preliminary experiment

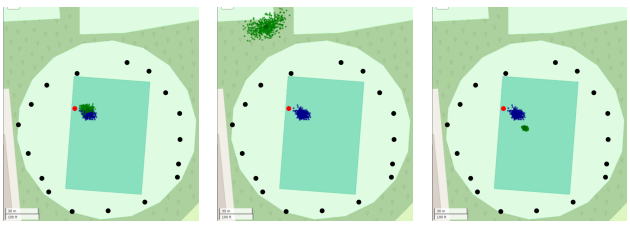
To evaluate the performance of the presented localization methods, a DRI receiver network of seventeen ESP32-C3s is used for the localization experiment. Each ESP32-C3 is equipped with 2.15dBi stacked dipole whip antennas [27] and is distributed in a circular area with known positions. One of the receivers can be seen on Fig. 10. For each experiment, 500 DRI messages are transmitted from a static known location. Three different experiments are conducted; 1) static RSSI localization experiment with its default transmissions power, 2) static RSSI localization experiment with different transmissions power, and 3) RSSI localization experiment while moving the spoofing device around.

Static RSSI localization experiment: The static RSSI localization experiment is performed in 15 different locations. The results from three of these can be seen on Fig. 11.



Figure 11: Receiver locations are in black, and the true transmitter location is in red. WCWCL estimates ($g = 1.5$) are shown in blue, and multilateration estimates ($n = 2$) in green. The mean estimate location is marked with larger green and blue icons.

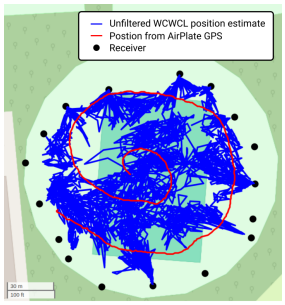
Static RSSI localization experiment with changing transmission power: Assuming a spoofing device was present, a continuously varying transmitter power could be expected to simulate different locations. To simulate a transmitter device changing its output power, the RSSI measurements are manually offset with $\pm$5dB, as shown in Fig. 12.



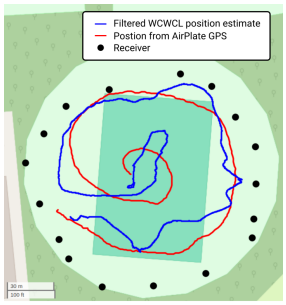
(a) Original RSSI measurements (b) -5dB from RSSI to simulating weaker transmitter (c) +5dB to RSSI simulating stronger transmitter

Figure 12: RSSI localization using whip antennas, showing the impact of changing transmitter power. Receiver locations are in black, true transmitter locations in red, WCWCL estimates ($g = 1.5$) in blue, and multilateration estimates ($n = 2$) in green. The mean location of estimates is marked with larger green and blue icons.

Moving RSSI localization experiment: A final localization experiment was conducted while moving an AirPlate around. The experimental setup is identical to the previous two static experiments. The AirPlate is moved around in a spiral for 398 seconds while the position is logged through the transmitted DRI messages. The WCWCL position estimate and the global position from the AirPlate are plotted on Fig. 13a. A moving average with a window size of 200, corresponding to 20 seconds of data, is used to filter the position estimates as shown on Fig. 13b.



(a) Unfiltered WCWCL position estimate.



(b) Moving average of WCWCL position estimate.

Figure 13: Comparison of the logged GPS position of the AirPlate in red and the estimated position using WCWCL with $g = 1.8$ in blue. The receiver locations are visualized in black.

3.3.2 Real-world validation

To evaluate the practicality of the proposed solution in a real-world context, an experiment was conducted at the Danish national UAS test center, at Hans Christian Andersen Airport (HCAA), Odense, Denmark, where spoofing poses a potential security threat. For this experiment, eight DRID receivers configured to log DRID messages onto an SD card were deployed in a triangular arrangement on a small section of the airport fence, with a maximum separation of 150 meters between them. All receivers were equipped with 9dBi stacked dipole antennas. The DRID emitter used in this experiment was an AirPlate [22], which was mounted underneath a DJI Mini 4 Pro drone. The drone was flown around the airport at various altitudes to simulate different operational conditions. Figure 14 below illustrates the experimental setup.

To test the localization accuracy of the system, the drone was hovered for 15 seconds at altitudes of 25, 50, 75, and 100 meters, respectively. The results of those hover tests are presented in Table 2, and can be visualized in Figure 15. It can be seen from Table 2 that the mean error tends to increase at higher altitudes.

Additionally, the UAV was flown around the airport at a constant altitude to observe the system's behavior as the UAV crossed the boundary of the coverage area. As shown in Figure 16, when the UAV operates outside the coverage area,



(a) DRID receiver with 9dBi stacked dipole antenna, and SD logging capability.



(b) AirPlate DRID emitter mounted underneath the DJI Mini 4 Pro.



(c) Map of the location of the DRID receivers at the HCAA.

Figure 14: Real-world spoofing localisation experimental setup at HCA airport using eight DRID receivers, and an AirPlate DRID emitter mounted below a DJI Mini 4 Pro.

Table 2: Mean Error and Standard Deviation of Hover Test

Altitude	Mean Error	Standard Deviation
25 m	5.026 m	17.902 m
50 m	9.217 m	39.380 m
75 m	41.900 m	17.788 m
100 m	13.322 m	24.225 m

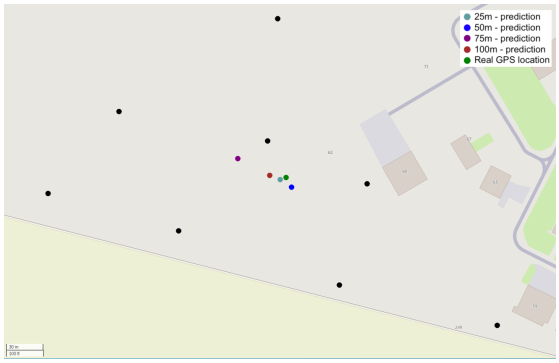


Figure 15: Localisation results from the experiment at altitudes of 25, 50, 75, and 100 meters. The colored dots represent the averaged localization predictions from the system, while the green dot indicates the true GPS location of the UAS at each altitude.

http://www.imavs.org/

the predicted location is positioned at the edge of the coverage zone, leading to inaccurate localization. However, this prediction can still serve as a directional guide, assisting in narrowing down the search area outside the fence.

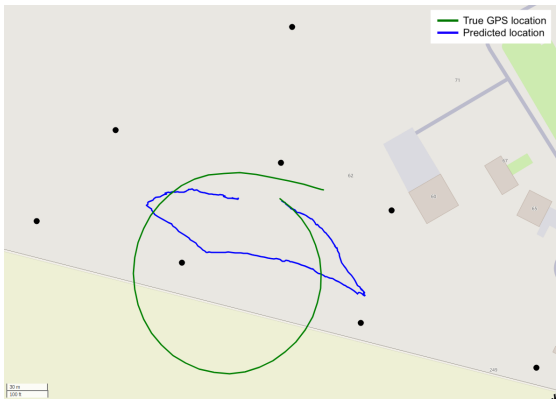


Figure 16: Localisation results as the UAS flies in a circle at a constant altitude crossing the boundary of the coverage area.

4 DISCUSSION

Theoretical Limits of the Broadcast Algorithm: The theoretical upper and lower bounds on the maximum number of nodes in a simple flooding mesh network were estimated based on idealized assumptions: perfect synchronization, no collisions, no additional network traffic, no latency, and a static topology. With a transmission time of 1.674 ms per message [20] and a broadcast rate of 1Hz, the maximum number of nodes in a fully connected dense network is estimated to be 24. For a sparse network with nodes in a straight line, the maximum is 199. These calculations illustrate the redundancy in transmissions within a dense network using simple flooding.

Comparison Between Multilateration and WCWCL: The WCWCL algorithm outperforms multilateration, with a mean error of 27 m compared to 35.8 m. Multilateration is also more computationally intensive due to least squares minimization. However, WCWCL cannot localize devices outside the convex hull formed by the receiver nodes, limiting the detection area for spoofing. WCWCL's robustness to changes in transmit power, as shown in Fig. 12, is a significant advantage, unlike multilateration, which is sensitive to transmitter power changes. Both methods are only effective for devices using omnidirectional antennas; directional antennas require a different approach and other algorithms.

Error sources: The localisation experiments were conducted in an unobstructed environment. Any obstructions affecting signal strength will impact location estimates.

Extended range of DRI: DRI messages were successfully transmitted through the FANET over a 1.6 km range. This range is not a limitation of the FANET; theoretically, DRI messages can be extended indefinitely if reliable connections exist between all nodes. However, the FANET implementation does not verify DRI message authenticity, potentially allowing a spoofing device to extend its coverage if within FANET range.

Message Frequency Through the FANET: Fig. 8 shows the frequency at which DRI messages are received by the scanner through the FANET. For the AirPlate, the frequency is ≈ 0.6Hz, which does not meet the DRI standard, although the AirPlate complies with the 1Hz regulation. Message frequency decreases with the number of hops through the network. Reliability can be enhanced by increasing rebroadcasts or the original transmission frequency, as discussed in [20].

Location Estimate of Moving Versus Static Transmitter: The moving transmitter experiment yielded better results than the static one, likely due to a bias-variance tradeoff. Static transmitters might introduce a bias due to misalignment, while the movement of the AirPlate reduces bias but increases variance, which can be mitigated through filtering.

5 CONCLUSION AND FUTURE WORK

To extend the range of DRI messages using a FANET, different broadcast algorithms were evaluated, and simple flooding was found to be the most effective for sparse and dynamic mesh networks. The results confirm that the implemented algorithm effectively extends DRI message range. DRI messages from a quadcopter with an original frequency of 10 Hz were received at 1.5 Hz to 8 Hz, while messages from the AirPlate transmitted at 1 Hz were received at approximately 0.6 Hz. The experiment also confirmed successful DRI and Mavlink integration.

For DRI message spoofing, two RSSI localisation methods were tested. The WCWCL method proved more favourable than multilateration, as it is robust to changes in transmitter power and does not require reference measurements. Although both methods showed significant localisation errors, WCWCL provided a reasonable estimate of a moving DRI transmitter using a moving average.

Future work includes: 1) adding fields to the DRI payload for hop counter, RSSI, location from the first forwarding node, and authentication signatures; 2) equipping a UAV with an ADS-B receiver to enhance awareness of nearby aircraft; 3) extending localization to 3D by arranging the receiver network in different planes; 4) implementing a hybrid localization approach with both omnidirectional and directional antennas to allow WCWCL to determine if a transmitter is outside the receiver network's convex hull; and 5) developing a self-calibration method where receiving antennas routinely broadcast fixed-power messages to improve accuracy and reduce variance from external disturbances like weather.

ACKNOWLEDGEMENTS

The authors thank August Ravn Mader and Troels Dupont Andreasen from Airplate [22] for their collaboration and for providing the DRI device used in the experiments. This work was supported by the **WildDrone MSCA Doctoral Network**

http://www.imavs.org/

[28] funded by EU Horizon Europe under grant agreement #101071224 and the 5G ENabled communication Infrastructure for Unmanned Aerial System (GENIUS) project [25].

REFERENCES

[1] May 2019. Commission Implementing Regulation (EU) 2019/947 of 24 May 2019 on the rules and procedures for the operation of unmanned aircraft, http://data.europa.eu/eli/reg_impl/2019/947/oj/eng.

[2] Aerospace series - unmanned aircraft systems - part 002: Direct remote identification. Technical report, European Comittee For Standardization, 2023.

[3] ASD-STAN. Introduction to the European UAS Digital Remote ID Technical Standard.

[4] Internet Engineering Task Force Drone Remote ID Protocol.

[5] Drone Remote ID Protocol. datatracker.ietf.org/group/drip/about/.

[6] Christopher Ho et al. Flooding for reliable multicast in multi-hop ad hoc networks. In *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*, pages 64–71, Seattle Washington USA, August 1999. ACM.

[7] Sze-Yao Ni et al. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 151–162, Seattle Washington USA, August 1999. ACM.

[8] Wei Peng. On the reduction of broadcast redundancy in mobile ad hoc networks. In *2000 First Annual Workshop on Mobile and Ad Hoc Networking and Computing. MobiHOC (Cat. No.00EX444)*, pages 129–130, Boston, MA, USA, 2000. IEEE.

[9] A. Qayyum et al. Multipoint relaying for flooding broadcast messages in mobile wireless networks. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, pages 3866–3875, Big Island, HI, USA, 2002. IEEE Comput. Soc.

[10] Wei Peng. AHBP: An efficient broadcast protocol for mobile Ad hoc networks. *Journal of Computer Science and Technology*, 16(2):114–125, March 2001.

[11] Brad Williams. Comparison of broadcasting techniques for mobile ad hoc networks. In *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, MobiHoc '02, pages 194–205, New York, NY, USA, June 2002. Association for Computing Machinery.

[12] Asma Mesmoudi et al. Wireless Sensor Networks Localization Algorithms: A Comprehensive Survey. *International journal of Computer Networks & Communications*, 5(6):45–64, November 2013.

[13] Guangjie Han et al. Localization algorithms of Wireless Sensor Networks: a survey. *Telecommunication Systems*, 52(4):2419–2436, April 2013.

[14] Tashnim J.S. Chowdhury et al. Advances on localization techniques for wireless sensor networks: A survey. *Computer Networks*, 110:284–305, December 2016.

[15] Bo Yang et al. A Novel Trilateration Algorithm for RSSI-Based Indoor Localization. *IEEE Sensors Journal*, 20(14):8164–8172, July 2020.

[16] Xiuyan Zhu. RSSI-based Algorithm for Indoor Localization. *Communications and Network*, 05(02):37–42, 2013.

[17] J. Blumenthal et al. Precise Positioning with a Low Complexity Algorithm in Ad hoc Wireless Sensor Networks. *PIK - Praxis der Informationsverarbeitung und Kommunikation*, 28(2):80–85, June 2005.

[18] Quande Dong. A Novel Weighted Centroid Localization Algorithm Based on RSSI for an Outdoor Environment. *Journal of Communications*, 9(3):279–285, 2014.

[19] Seeed Studio. https://wiki.seeedstudio.com/XIAO_ESP32C3_Getting_Started/, 2024-06-13.

[20] Høffer, Mads Fogh and Herlev, Victor Dahl. Extending direct remote id reception coverage through ad-hoc mesh networking to improve airspace situational awareness. Master's thesis at University of Southern Denmark.

[21] MAVLink Developer Guide, https://mavlink.io/en/.

[22] AirPlate, https://airplate.dk/.

[23] PX4, https://px4.io/.

[24] QGroundControl. http://qgroundcontrol.com/.

[25] GENIUS. "https://genius.aero/".

[26] DRI relay experiment QGroundControl view. https://youtu.be/CdmBHrqAA-k.

[27] W1039B030 Datasheet. Pulse electronics, https://productfinder.pulseelectronics.com/api/open/part-attachments/datasheet/W1039B030.

[28] WildDrone, https://wilddrone.eu/.

http://www.imavs.org/

Effects of Simultaneous Turbulence and Ice Accretion on Small Fixed-Wing UAV Performance

T. Y. Ma*, N. Carlier, N. Kay†,
Department of Mechanical and Mechatronics Engineering, University of Auckland, Auckland, New Zealand

A. Pirooz,
National Institute of Water and Atmospheric Research, Auckland, New Zealand

C. Azorin-Molina
Spanish National Research Council (CIDE-CSIC), Valencia, Spain

http://www.imavs.org/

ABSTRACT

Small Unmanned Aerial Vehicles (UAVs) are often required to operate in unfavourable weather conditions, such as unsteady turbulent winds and potentially in icing conditions. While the effects of turbulence and icing on small UAVs have been considered individually, little has been said of their combined effect on flight performance, endurance, and so the consequent probability of successfully completing the operation. As such, this work looks at the overall impact of simultaneous ice-accretion and turbulence conditions on aircraft performance for a fixed-wing Kahu UAV via wind tunnel testing. It was observed that ice has surprisingly little effect on the aerodynamic forces produced by the Kahu UAV, with one ice condition enhancing the lift-to-drag ratio. In contrast, increasing turbulence intensity significantly altered all forces and moments. The greatest change due to icing was seen in the roll moment variability. It is concluded that Kahu can be operated in icing conditions, albeit with a loss of endurance due to the excess power required.

1 INTRODUCTION

The utility of small UAVs has expanded rapidly, due to the development of small, lightweight electronics [1, 2]. Courtesy of their small size, these aircraft are able to operate in "dull, dirty and dangerous" environs which are either too hazardous for conventional aircraft, or not financially viable for the latter due to the supporting infrastructure required [3]. While multirotor configurations are favoured for short-range operations requiring hovering, fixed-wing designs, typically up to 2 m in span, offer range and endurance at low cost [4]. Although these may look like ordinary aircraft on a smaller scale, they differ aerodynamically through their much lower Reynolds numbers, often less than 250,000, and the consequent low-energy laminar boundary layer.

With regulations and roles often forcing small, UAVs to low altitudes, these aircraft are exposed to the highly unsteady flows seen low in the Atmospheric Boundary Layer (ABL). This combines with the typically-low flight speed of these aircraft to produce high onset turbulence intensities, reaching as high as 30% in urban operating conditions. The combination of low Reynolds number aerofoils has been found to result in significant unsteady flow separation, producing transient loads akin to dynamic stall. The mean lift coefficient at stall, however, can reduce by as much as 30%, while stall occurs at a higher Angle of Attack (AoA) [5]. This provides a significant challenge for UAV operations.

Icing is also a major challenge for research and rescue operations, when small UAVs are required to work in cold environments. As with turbulence, icing has a disproportionate effect on small UAVs due to their relatively low mass compared to the ice weight [6]. Aerodynamically, ice has a significant effect due to the sensitivity of the boundary layer at low Reynolds numbers. Hann and Johansen [6] noted that not only does stall occur at a lower AoA, with the loss of lift reaching 30%, but also drag doubles from the baseline value. Additionally, de-icing equipment is rarely fitted due to the limited mass and power capacity available onboard the aircraft [7].

In operation, a small UAV may encounter both turbulence and icing. While studies have been conducted for the individual effects of these environmental factors, the combination of the two has received little attention. Therefore, this work expands on the current knowledge base by looking at the impact of ice on the performance of a small UAV in turbulence.

2 PHYSICAL TESTING

The aircraft employed in this work is a fixed-wing Kahu UAV. This aircraft has been used for prior icing research, notably by Williams et al. [8]. It has a span of 2.316m, operating mass of 3.5 kg, and a cruising speed of 16.7 m s^{-1} (60 km/h). This results in a typical Reynolds number (based on the mean aerodynamic chord) around 230,000. The aircraft

* trista.ma@auckland.ac.nz
† nicholas.kay@auckland.ac.nz

http://www.imavs.org/

can be seen in Figure 1 in the University of Auckland Boundary Layer Wind Tunnel.

Aerodynamic forces acting on the aircraft were recorded via a JR3 160M50A4-180-EF six-axis analogue loadcell for 60 seconds at a frequency of 1000 Hz. The data of interest were the lift, drag and pitch moments, as well as any changes in the roll moment. These have been presented as coefficients, as represented in Eqs. 1 to 4, for the lift (L), drag (D), pitch (M) and roll moments (R). Note the q is the dynamic pressure, S the reference wing area (0.484 m), c is the mean wing chord (0.209 m) and b is the wing span. Data was recorded in 2° Angle of Attack (AoA) increments from -10° to +20°, based on the wing root chordline. Force and moment offsets due to the stand were subtracted prior to analysis. With the test section 3.6 m wide and 2.5 m tall, the maximum blockage was 4%, and so velocity corrections were not applied.



Figure 1: Kahu UAV on load cell, with turbulence grid upstream

$$C_L = L/qS \quad (1)$$

$$C_D = D/qS \quad (2)$$

$$C_M = M/qSc \quad (3)$$

$$C_{Roll} = L/qSb \quad (4)$$

To simulate atmospheric turbulence, a passive, biplane turbulence grid was placed upstream of the aircraft as required, which can be seen in Figure 1. By varying the distance between the grid and Kahu, the turbulent flow conditions listed in Table 1 were produced. The primary parameter varied was the turbulence intensity. Although less than seen in an urban environment, this is still significant in the context of UAV operations. The use of this passive grid restricted the wind tunnel speed to 10 m s^{-1}, due to back pressure on the axial-flow fans. This resulted in an experimental mean-chord Reynolds number of 140,000. While less than ideal, the elevated turbulence intensity was anticipated to mitigate the difference in Reynolds number.

The ice shapes used in this work were those of Williams et al. [8]. These were generated on an RG-15 wing section (as used by Kahu) in an icing wind tunnel to the conditions of FAR Part 25 Appendix C. These represent cruising at an altitude of 1.5 km in an ambient -20°C, with a liquid water content of 1.9 g m^{-3}. Williams et al. [8] defined this as an "Intermittent Maximum" (IM) condition, and so each ice shape is designated by this and the AoA at which it was formed in Table 2. These profiles were extruded along the wing form of Kahu and 3D printed for wind tunnel testing. Ice on the tail unit and fuselage was not considered. The mass of each ice shape was determined from the CAD and added to the baseline net aircraft mass for performance assessment. This was conducted for each combination of ice shape and onset turbulence.

Turbulence Intensity, I_u	Length Scale, L_{xx}
1.1%	0.56 m
5.6%	0.32 m
8.0%	0.44 m
13.3%	0.42 m

Table 1: Turbulence Characteristics

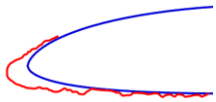
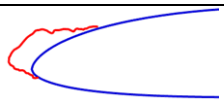
State		**Aircraft Mass (kg)**	**Form**
No Ice		3.50	
Ice	0° IM	3.60	
	+4° IM	3.66	
	-4° IM	3.66	

Table 2: Ice shapes and aircraft net mass

3 RESULTS

3.1 Time-Averaged Forces and Moments

Considering time-averaged data first, Figure 2 shows the clean Kahu airframe mean lift-to-drag ratio to be relatively insensitive to turbulence, maintaining a maximum C_L/C_D of 15.3, between 2° to 4° AoA for all turbulence intensities. While the addition of ice does not change the AoA of the maximum C_L/C_D, it does change the magnitude, as well as sensitivity to turbulence. This is best seen in the +4° IM and -4° IM cases. For the former, the maximum C_L/C_D increases to 16.3 at the lowest turbulence intensity, rising to 19.2 at the highest. This shape is seen in Table 2 to be the most conformal to the aerofoil geometry, and provides a small cambered leading edge extension. The -4° IM case, in contrast, lowers the maximum C_L/C_d to 13.2 when I_u = 1.1%, rising to C_L/C_d to 15.7 when I_u = 13.3%. This reduced aerodynamic performance can also be deduced from the form of the ice shape, which decambers the leading edge as a prominent upper-surface feature.

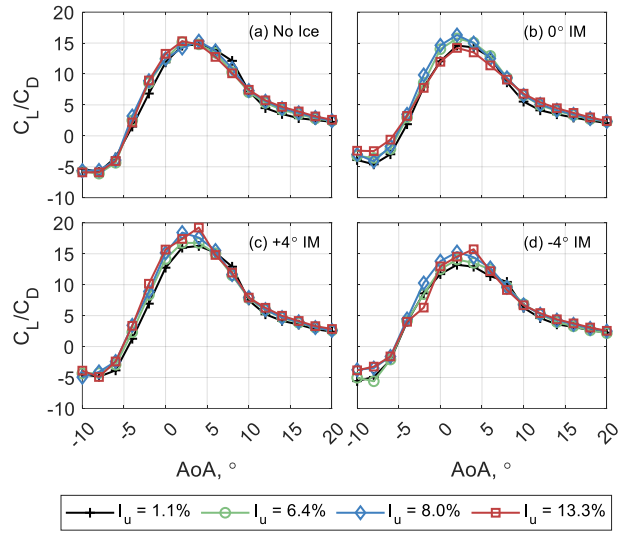


Figure 2: Time-averaged Lift-to-Drag Ratio for each ice configuration at all turbulence intensities.

Figures 3 and 4 resolve the lift and drag coefficients individually. The maximum lift coefficient for all cases can be seen in Figure 3 to initially rise with turbulence intensity, but reduces at the highest I_u as the lift slope declines. The AoA for maximum lift also increases with I_u. This is consistent with findings for reference aerofoils under similar conditions [5]. However, regardless of turbulence intensity, all ice shapes produce a more gradual stall than the baseline wing. This may be due to the ice shapes forcing an earlier boundary layer transition, and so delaying separation [9]. Oo et al. [10] observed aa laminar separation bubble for the Kahu wing in CFD with no onset turbulence, whereas ice promoted attached flow. This difference is mitigated at higher turbulence intensities, when laminar flow is unlikely to occur on the baseline wing [11].

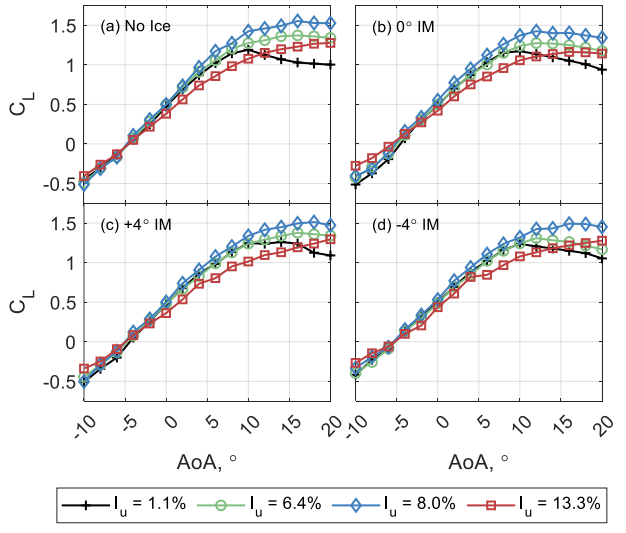


Figure 3: Time-averaged Lift Coefficient for each ice configuration at all turbulence intensities.

Unexpectedly, there is relatively little difference between the ice shapes and the clean wing, albeit with the +4° IM producing a slightly delayed stall. This is in contrast to the findings of Hann and Johansen [6] and Cheng et al. [12], who saw a reduction in the maximum lift coefficient exceeding 10%. The latter study, however, does not state the onset turbulence intensity. The baseline in the present work is 1.1%: this would be a significant factor in comparison to a smooth onset flow. Furthermore, Cheng et al. [12] use the effective wing area, including the change in reference planform due to the ice extending beyond the wing leading edge. Williams et al. [8] also note this as a factor for reducing the prominence of the ice, but also observe that it is impractical to change wing areas for different cases. In the present study, the ice would increase the wing area by 2.4%, 2.7% and 2.6% for the 0° IM, +4° IM and -4° IM, respectively.

Comparing to an experimental study with a similar baseline turbulence intensity, Oo et al. [10] recorded greater deviations in force due to the same ice shapes on the same wing. The difference, however, is that the aforementioned study looked solely at the wing and projected performance changes on the entire aircraft. The present work includes the entire airframe. Therefore, other sources of lift, notably the tailplane, are also included in the recorded data. This would dilute the impact of the ice, which is only applied to the wing.

As with the lift coefficient, Figure 4 shows that the primary difference in drag is evident around stall. While there is a slight reduction in drag at low AoA with increasing I_u, but an increase with ice, the overall form and magnitude are comparable. Above 10°, however, there is a significant increase in drag at the lowest turbulence intensity, the slope reducing again beyond 16°. As the turbulence intensity increases to 5.6% and 8.0%, this latter reduction in slope does not occur, the drag coefficient instead continuing to rise. This aligns with the findings of Tran et al. [13] for an isolated Kahu wing. On the other hand, at the highest turbulence intensity, the initial increase in drag is flattened out. The reason for this is not immediately clear, although this may be due to the delayed stall observed in the lift.

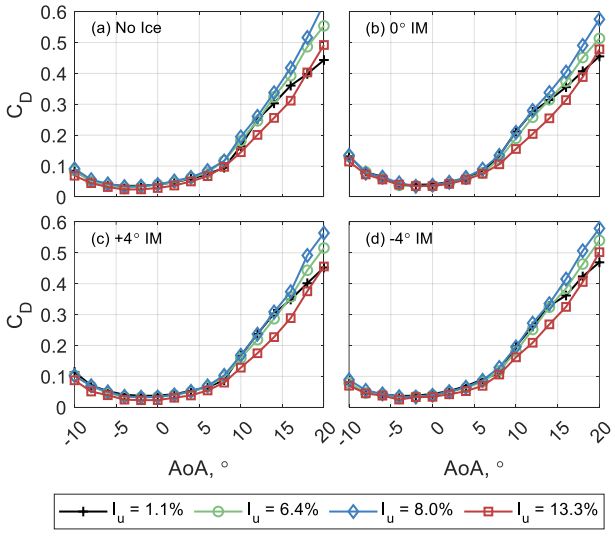


Figure 4: Time-averaged Drag Coefficient for each ice configuration at all turbulence intensities.

http://www.imavs.org/

http://www.imavs.org/

The moment coefficient, seen in Figure 5, displays more variability with ice, particularly at the highest turbulence intensity. Note that the high magnitude of the nose-down pitch moment at high AoA is due to the lift generated by the tailplane, cited 0.56 m aft of the CoG. The primary difference is the more nose-up moment of the 4° IM condition at AoA below 10°, which then becomes more negative than the baseline above this AoA at the lowest turbulence intensity. As the latter increases, a more nose-up moment is produced for all ice shapes. Given that the magnitude of the pitch moment coefficient remains within the bounds of what is seen with the clean wing, this indicates that Kahu would still have sufficient pitch authority in the absence of turbulence.

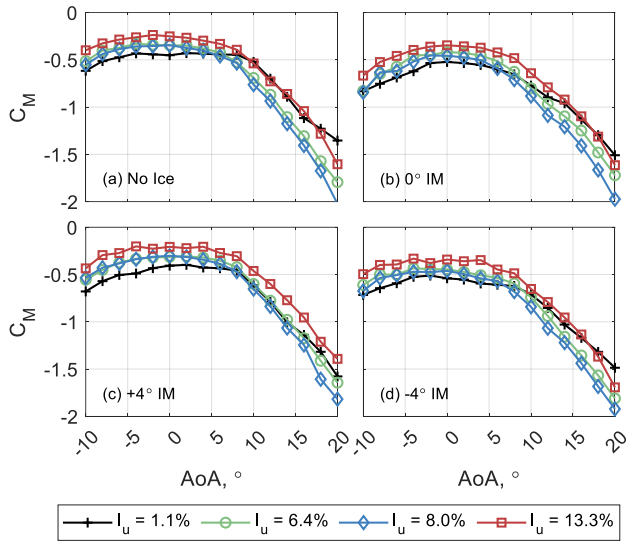


Figure 5: Time-averaged Pitch Moment Coefficient for each ice configuration at all turbulence intensities.

3.2 Unsteady Loads

Due to the highly unsteady onset flow, the resulting forces acting on the aircraft are also expected to be increasingly unsteady. The standard deviation is used to quantify this unsteadiness, and can be seen in Figures 6, 7, and 8 for the lift, drag and pitch moment coefficients, respectively. As with the time-averaged cases, the onset turbulence dominates the effect of the icing. This notably is seen primarily between $I_u = 1.1\%$ and $I_u = 5.6\%$, subsequent increases in turbulence intensity producing lesser changes in the unsteady loads. The magnitudes for the clean wing and all iced conditions are, in general, similar. Again, this is likely due to the effect of the rest of the airframe, with the tailplane providing a significant contribution to the pitch moment in particular.

Where differences can be seen is around stall. Whereas a sharp rise in unsteadiness can be seen for the iceless case, all iced conditions see a more gradual increase for each load. This may be due to promoting localised separation of the shaper leading edge earlier, leading to transient vortex shedding [14] and so greater force variability at a lower AoA. The 0° IM case also shows a notable increase in σ_{C_D} relative to other cases when $I_u = 15\%$. The reason for this is unclear: there is no corresponding increase in the standard deviation of the lift and pitch moment coefficients, which suggests it is not related to mass flow separation. Local separation has been noted to cause high-frequency unsteadiness in the pressure coefficient [15], but this has only been observed at low turbulence intensities. as with the laminar separation bubble, these effects would be unlikely to be evident as the onset flow becomes more unsteady and inhibits the maintenance of a laminar boundary layer.

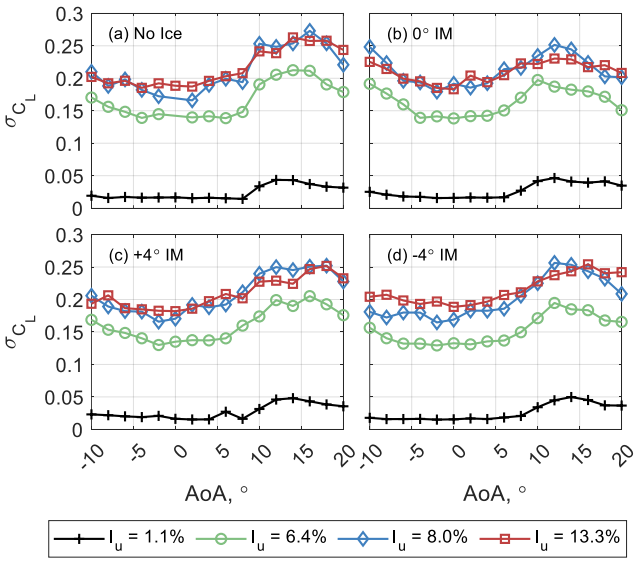


Figure 6: Standard deviation of the Lift Coefficient for each ice configuration at all turbulence intensities.

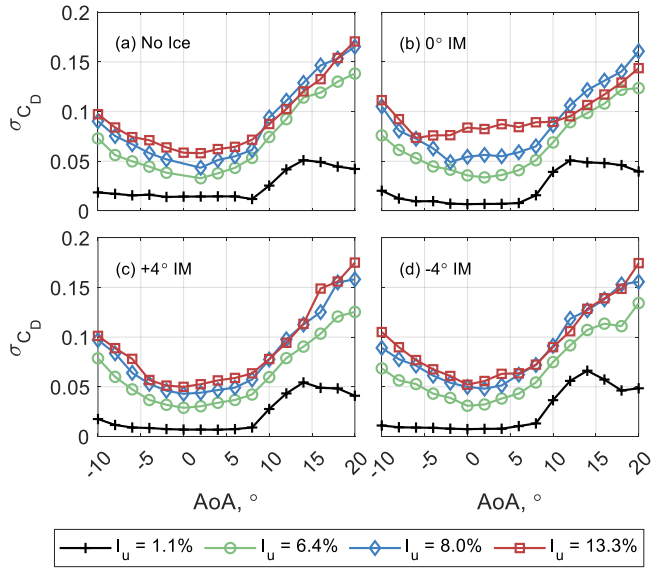


Figure 7: Standard deviation of the Drag Coefficient for each ice configuration at all turbulence intensities.

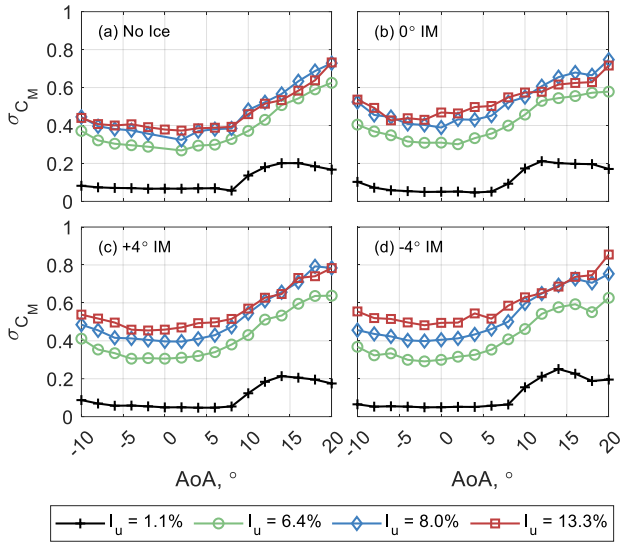


Figure 8: Standard deviation of the Pitch Moment Coefficient for each ice configuration at all turbulence intensities.

Additionally, the roll moment coefficient is presented in Figure 9. The time averaged mean was not shown as this is zero for all AoA (Kahu being restrained, symmetrical and oriented into the wind). However, Marino et al. [16] observed that roll deviation was the greatest concern for managing flightpath deviation and stable photography. At low flight speeds, the spanwise coherence of turbulent flow conditions is reduced, increasing susceptibility to roll disturbances [17].

As expected, increasing the turbulence intensity increases $\sigma_{C_{Roll}}$. Again, this is most prominent between the lower turbulence intensities, higher intensities producing little difference for the clean airframe. There is also an increase at stall, local vortex shedding occurring as the flow separates from the wing.

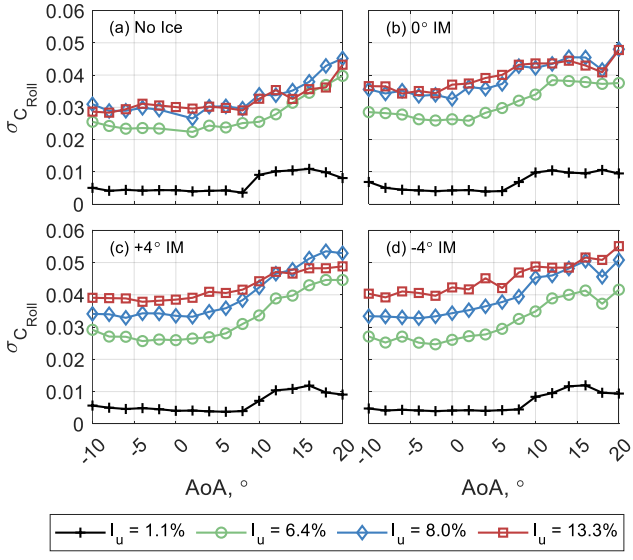


Figure 9: Standard deviation of the Roll Moment Coefficient for each ice configuration at all turbulence intensities.

However, the ice shapes do show a difference, generally inducing a higher roll moment variation at higher turbulence intensities for all AoA. This may be related to the ease of separation expected to be induced by the sharper leading edge [14]. Roll is more sensitive than the lift and drag coefficients in this regard due to the moment arm involved. As different flow conditions are seen along the span of the wing, local vortex formation and flow separation can occur simultaneously at different distances from the centre of gravity. For net forces, some of these local effects average out across the span. For a moment, however, this local variation can add together to produce a net increase. As such, the effect of the ice become more notable for roll, with its long offset to the centre of gravity. The pitch moment is less impacted due to the short moment arm of the wing in this axis.

3.3 Flight Performance

Two aspects are looked at for observing the impact of the ice and turbulence on overall flight performance: minimum flight speed and excess power required. The former can be seen in Table 3. Due to the unsteady nature of the forces generated, this speed is taken with the maximum 5th percentile lift. Use of the maximum time-averaged lift would result in significant periods of insufficient lift to fly, the wing intermittently stalling [5], while use of the minimum would be overly-conservative due to noise in the peak loads. As propulsive efficiency curves are not available, the vertical contribution of motor thrust is neglected, but would slightly decrease this speed.

I_u	No Ice	0° IM	+4° IM	-4° IM
1.1%	10.1	10.4	10.1	10.1
5.6%	10.6	11.0	10.7	11.0
8.0%	10.1	10.8	10.5	10.5
13.3%	11.4	12.1	11.4	11.6

Table 3: Minimum Flight Speed based on maximum 5th percentile lift, m s^{-1}

The presence of ice has minimal impact on the minimum speed. While this is surprising, given the extra mass due to the ice, the increase in the lift generated at higher AoA offsets this. In contrast, increasing the onset turbulence intensity increases the flight speed required, primarily due to the increasing spread of lift coefficients produced. This is most apparent at the highest turbulence intensity, where not only is the spread of transient lift greater, but also the maximum mean lift coefficient produced has reduced.

The excess power required can be seen in Table 4. This is the power difference from the baseline case to maintain flight with best endurance as the target. As such, the minimum power required was calculated for each mass and set of aerodynamic data, and the difference found from the baseline. With the exception of +4° IM at the lowest turbulence intensity, any ice or increase in turbulence intensity produces a greater power requirement. This is again

http://www.imavs.org/

dominated by turbulence, reflecting the greater power needed to maintain safe flight. The increase in power for the ice primarily comes from the greater AoA needed to offset the weight, and as a consequence an increase in drag. The +4° IM case has the benefit of its mild increase on both area and camber producing a higher lift-to-drag ratio.

I_u	No Ice	0° IM	+4° IM	-4° IM
1.1%	0.00	2.94	0.00	6.65
5.6%	11.22	15.54	11.75	18.29
8.0%	12.26	20.35	13.24	19.02
13.3%	36.47	48.13	28.72	30.56

Table 4: Excess power required at minimum power velocity, W

With a 1.2 kW motor, it is unlikely that that Kahu would be unsafe to operate in such conditions, and overall Kahu seems to be more resilient to icing than expected. However, these power increases would result in a loss of endurance. Previous work found that Kahu typically consumes around 100 W [18] in flight with all systems operational. Taking this value, the endurance loss in the highest turbulence intensity with the 0° IM ice shape would be as much as 48%. Flying slightly nose-up into adverse conditions, as per the +4° IM case mitigates this.

4 DISCUSSION

The present work looked only at wing icing due to the icing data available. However, ice is also known to significantly impact propeller aerodynamics. This would reduce power output, and hence the margin available to offset the added power requirements due to ice and turbulence.

The impact of tailplane icing has also not been considered. It is unclear how much ice would form on the tailplane, sited in the wake of the propeller. Scaling the ice shapes used here to the tailplane geometry produces a net added mass of 7.9 g, or a nose-up moment of 4.7 Nm when flying with a horizontal attitude (maximum moment arm). In terms of the pitch moment in Figure 5, this would be a negligible change in C_M of 7.5×10^{-4}. However, the impact on airframe drag or on tailplane control authority cannot be estimated in such a manner. Kahu has an all-moving tail unit: the effects of ice on the wing would suggest that sufficient AoA range would remain available for control to be effective. To obtain a better picture, ice should be generated on the whole Kahu airframe with a variety of power settings and tailplane rotations.

The impact of icing on control surface effectiveness was also not considered, as the wind tunnel model does not have operable control surfaces. It would be of interest to assess the changes in effectiveness of each surface in the presence of the upstream disturbances.

Finally, a major constraint in wind tunnel testing with turbulence is the inability of the UAV to move with the flow. In reality, the aircraft does not see the full gust variability as it is displaced by the gust. This can result in overestimation of peak loads, and so a conservative decision on whether it is safe to fly. While this is better than the converse situation, it does remain a limitation on the applicability of the wind tunnel data.

4. CONCLUSIONS

While both icing and turbulence impact the aerodynamic performance of the Kahu UAV, generally the effect of turbulence dominates that of the ice. The primary instance when the impact of ice is evident is towards stall. Surprisingly, one of the icing conditions improved the lift to drag ratio, likely through suppression of laminar separation in addition to increasing the effective wing area and camber. Thus, the effects of icing and turbulence are minimised if the UAV is flown into icing conditions with a slight nose-up attitude. Due to the propulsive power available on Kahu, it is unlikely that cancellation of the flight would be necessary, although the endurance would be decreased.

ACKNOWLEDGEMENTS

This research was conducted as a Final Year Research Project (BE(Hons)) in the Boundary Layer Wind Tunnel at the University of Auckland.

This study forms part of the ThinkInAzul programme and was supported by MCIN with funding from European Union NextGenerationEU (PRTR-C17.I1) and by Generalitat Valenciana (THINKINAZUL/2021/018).

REFERENCES

[1] S. Aggarwal and N. Kumar, "Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges," *Computer communications,* vol. 149, pp. 270-299, 2020.

[2] J. M. McMichael and M. S. Francis, "Micro air vehicles-toward a new dimension in flight," DARPA, USA, 1997.

[3] R. Austin, *Unmanned aircraft systems: UAVS design, development, and deployment*, 1 ed. (no. Book, Whole). West Sussex, UK: John Wiley & Sons, 2010.

[4] P. Fahlstrom and T. Gleason, *Introduction to UAV Systems*, 4 ed. (Introduction to UAV systems, no. Book, Whole). West Sussex, UK: John Wiley & Sons, 2012, pp. 17-31.

[5] N. J. Kay, P. J. Richards, and R. N. Sharma, "Influence of Turbulence on Cambered and Symmetrical Airfoils at Low Reynolds Numbers," *AIAA Journal,* vol. 58, no. 5, pp. 1913-2325, 2020, doi: https://doi.org/10.2514/1.J058822.

[6] R. Hann and T. A. Johansen, "UAV icing: The influence of airspeed and chord length on performance degradation," *Aircraft Engineering and Aerospace Technology,* vol. 93, no. 5, pp. 832-841, 2021.

http://www.imavs.org/

[7] L. Zhou, X. Yi, and Q. Liu, "A Review of Icing Research and Development of Icing Mitigation Techniques for Fixed-Wing UAVs," *Drones,* vol. 7, no. 12, p. 709, 2023.

[8] N. Williams, A. Benmeddour, G. Brian, and M. Ol, "The effect of icing on small unmanned aircraft low Reynolds number airfoils," in *17th Australian International Aerospace Congress: AIAC 2017*, 2017, no. Conference Proceedings: Engineers Australia, Royal Aeronautical Society, p. 19.

[9] M. Istvan, J. Kurelek, and S. Yarusevych, "Effects of free-stream turbulence intensity on laminar separation bubbles," in *46th AIAA Fluid Dynamics Conference*, Washington D.C., USA, 2016, no. Conference Proceedings: AIAA, pp. 1-13, doi: https://doi.org/10.2514/6.2016-3946. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84980410219&partnerID=40&md5=c2da9bd9b85ffd99783396218b437394

[10] N. L. Oo, N. J. Kay, A. J. Brenkley, and R. N. Sharma, "Investigation into the behaviour of an iced low Reynolds number aerofoil," in *10th International Micro-Air Vehicles Conference*, Melbourne, VIC, Australia, Ruasrt, Ed., 2018 2018, no. Conference Proceedings: IMAV, pp. 174-179. [Online]. Available: http://www.imavs.org/papers/2018/IMAV_2018_paper_5.pdf. [Online]. Available: http://www.imavs.org/papers/2018/IMAV_2018_paper_5.pdf

[11] T. J. Mueller, L. J. Pohlen, P. E. Conigliaro, and B. J. Jansen Jr, "The influence of free-stream disturbances on low Reynolds number airfoil experiments," *Experiments in Fluids,* vol. 1, no. 1, pp. 3-14, 1983, doi: https://doi.org/10.1007/BF00282261.

[12] H. Cheng, D. Zhao, N. L. Oo, X. Liu, and X. Dong, "Numerical Investigation on Intermittent Maximum Ice Accretion and Aerodynamic Performances of RG-15 Aerofoil at Low Reynolds Number," *Aerospace,* vol. 11, no. 1, p. 7, 2023.

[13] H. B. N. B. Tran, P. J. Richards, and R. N. Sharma, "Unsteady Aerodynamic Response of the Kahu UAV Wing under Turbulent Conditions," in *AIAA AVIATION 2021 FORUM*, 2021, p. 2582.

[14] J. H. S. Fincham and M. I. Friswell, "Aerodynamic optimisation of a camber morphing aerofoil," *Aerospace Science and technology,* vol. 43, pp. 245-255, 2015, doi: https://doi.org/10.1016/j.ast.2015.02.023.

[15] N. L. Oo, P. J. Richards, and R. N. Sharma, "Ice-Induced Separation Bubble on RG-15 Airfoil at Low Reynolds Number," *AIAA Journal,* vol. 0, no. 0, pp. 1-12, doi: 10.2514/1.J059257.

[16] M. Marino, S. Watkins, R. Sabatini, and A. Gardi, "Unsteady pressure measurements on a MAV wing for the design of a turbulence mitigation system," in *Metrology for Aerospace (MetroAeroSpace)*, 2014 2014, no. Conference Proceedings: IEEE, pp. 138-143, doi: https://doi.org/10.1109/metroaerospace.2014.6865909.

[17] S. Watkins, J. Milbank, B. J. Loxton, and W. H. Melbourne, "Atmospheric winds and their implications for Micro Air Vehicles," *AIAA Journal,* vol. 44, no. 11, pp. 2591-2600, 2006, doi: https://doi.org/10.2514/1.22670.

[18] A. Gong, J. L. Palmer, G. Brian, J. R. Harvey, and D. Verstraete, "Hardware-in-the-loop simulation of a fuel-cell-based UAV propulsion system using real-world flight data," in *Proceedings of the Fourth Australasian Unmanned Systems Conference, Australian Association for Unmanned Systems*, 2014, p. 7.

http://www.imavs.org/

Gen4jectory algorithm – 4-D trajectory planning with minimised flight time for multiple rotary-wing UAVs

Ivan Panov*[*], Mouad Boumediene**, Henrik Skov Midtiby** and Kjeld Jensen**
* Department of Business Development and Technology, Aarhus University, Herning, Denmark 7400
** SDU Drone Center, University of Southern Denmark, Odense, Denmark 5230

http://www.imavs.org/

ABSTRACT

The paper proposes an innovative Gen4jectory algorithm that allows planning multiple 4-D trajectories considering unique aircraft performance data of the uncrewed rotary-wing air traffic. The algorithm guarantees zero cases of Loss of Separation (LoS). We describe the physical model of drone (Unmanned Aerial Vehicle (UAV)) motion, map representation, pathfinding technique, 4-D trajectory planning, check of LoS, experimental setup and simulation results. Finally, we discuss the promising ways of further research.

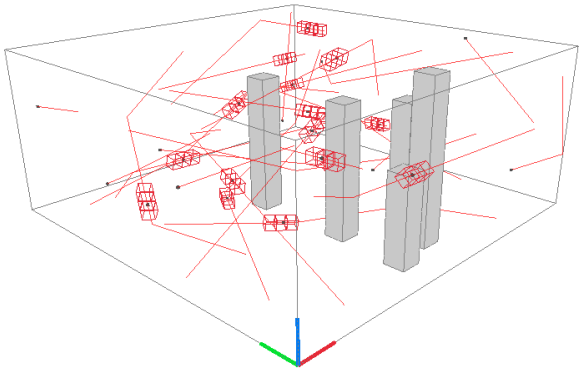


Figure 1: Visualization of the Gen4jectory algorithm in an urban environment

1 INTRODUCTION

The highest level of flight safety in Europe sets a top expectation on unmanned aviation integration in airspace [1]. Simultaneously, U-space and Urban Air Mobility stakeholders expect a high business efficiency of the uncrewed operations where autonomous systems will be able to substitute an expensive labour force. People have limited capabilities in manual planning and managing uncrewed air traffic in congested airspace. These limitations highlight the important need for an autonomous guidance system [2] which will be able to manage U-space and Urban Air Mobility traffic in a safe and efficient way.

However, what is the efficiency of the uncrewed operations? For business applications, the efficiency can be defined as on-time delivery rate, cost per mile, fuel efficiency, vehicle utilisation rate, etc. The metric choice depends on the operation requirements. For example, surveillance operations may require maximisation of endurance; consumer goods delivery and human transportation may require flight time minimization; operation in the night time may require noise limitation; heavy cargo delivery may require energy consumption minimization, etc. Practically speaking, it implies very different approaches to UAV 4-D trajectory planning. For instance, noise limitation could require higher altitudes of operation and avoidance of the noise-sensitive areas, flight time minimisation could require high tilt angles with a high level of thrust during UAV operation.

According to the forecast given by SESAR [3], delivery and transportation operations will cover about 45% of all UAV operations over Europe by 2050. We expect that delivery and transportation missions will require flight time minimisation in most cases. Bearing this in mind we decided to conduct a research project on uncrewed traffic 4-D trajectory planning focusing on the operations that require flight time minimisation.

4-D trajectory planning is a safety-critical aspect of autonomous UAV operations. It means that the proposed algorithm must exclude LoS for all airspace participants as a result of the operations planning.

The paper begins with an abstract and introduction. After that, we discuss our literature review and present the problem formulation. In the section “Algorithm design and methodology”, we discuss our physical model of a rotary-wing UAV, the choice of a map representation and pathfinding technique, and an approach for avoiding LoS. Then we discuss the simulation results, which is followed by a conclusion. A list of references finalizes the paper.

The main paper contribution is a novel Gen4jectory algorithm that allows planning 4-D trajectories for rotary-wing autonomous UAVs for autonomous uncrewed traffic management systems. (Figure 1. provides an example of 4-D trajectory planning in an urban environment (grey 3-D rectangles represent buildings, black points represent UAVs, and red 4-D rectangles indicate separation requirements).

Additionally, we make a comparison with a super sim-

*Email address: ivan.panov@btech.au.dk

plified model of flight, where flight time between waypoints is calculated by considering only maximum speed in level flight.

2 LITERATURE REVIEW

4-D trajectory planning for air traffic is a classical approach in the field of commercial aviation [4], [5]. However, this approach cannot be completely applied to the Unmanned aircraft system Traffic Management (UTM) as UAVs do flight with no flight corridors and flight levels. Additionally, 4-D trajectory planning for UTM requires a more complex approach as UAVs operate on a very low level facing ground obstacles and other on-surface objects and subjects.

4-D trajectory planning for commercial aviation considers aircraft performance which allows to pre-calculate with sufficient accuracy where and when the aircraft will be. In Europe, this issue is addressed via the Base of Aircraft Data (BADA) database, which incorporates essential aircraft performance data on certified aircraft [6]. The certification process of a commercial airliner is an expensive, well-documented process with an obligatory measuring of the aircraft performance data. There is no official demand for U-space air traffic participants to share aircraft performance data as a mandatory condition for operation approval. Therefore, 4-D trajectory planning for U-space is a complex task where alternative approaches must be identified or invented.

We expect that computational methods could help in obtaining quick and low-cost data extraction on aircraft performance based on the 3-D model of a UAV. However, we do not exclude machine learning algorithms that will be able to predict rough aircraft performance data based on a set of images of the UAV. We encourage the scientific community to contribute in this area to fill the gap.

While the issue of collecting aircraft performance data on UAVs remains unresolved, the researchers use assumptions or simply ignore aircraft dynamics constraints within their studies. For example, the authors of [7] proposed a solution for the multi-robot trajectory generation; however, the aerodynamic drag issue was not addressed. Similarly, the authors of [8] proposed the dynamic trajectory planning method missing an essential discussion on aircraft dynamics constraints. The authors of [9] invented a bio-inspired collision-free 4D trajectory generation method that incorporates a set of techniques to plan UAV trajectories like Tau-guidance, geometric, conflict detection, Particle Swarm Optimization, Conflict Detection, and Resolution. Again, the UAV dynamics model was not described in the paper. Similarly, the authors of the papers [10, 11, 12, 13] did not present a satisfactory physical model of the UAV to address 4-D trajectory planning.

In addition to the challenges in 4-D trajectory planning mentioned earlier, path planning in 3-D space for multiple UAVs also requires algorithms that can account for static obstacles such as buildings, and dynamic obstacles such as other UAVs sharing the same airspace. One such algorithm is theta-star [14] and its variants [15, 16], an extension of the A-star algorithm [17], which has been effectively used in 2-D path planning [18] and adapted for 3-D environments [19]. theta-star differs from A-star by allowing direct path connections (line of sight) between nodes, which reduces the path length and results in smoother routes [14].

Recent studies have explored the use of theta-star and its variants for UAV path planning in 3-D [20, 21], highlighting its ability to generate near-optimal paths while maintaining computational efficiency. For instance, theta-star has been applied in scenarios where UAVs must navigate urban environments with a high density of obstacles [22, 23]. This was done by considering the UAV's ability to move freely in three dimensions.

3 PROBLEM FORMULATION

A precise UAV flight dynamics model is a resolved issue [24]. However, precision requires precise aircraft performance data and costly computation time. For example, mass distribution on the UAV body is essential to estimate the corresponding moments. Obtaining such data for a large number of drone models is costly and inefficient. In this light, we decided to conduct a research project on a simplified flight dynamics model to allow 4-D trajectory planning for U-space needs. Additionally, we combined our flight dynamics model with the theta-star pathfinding algorithm [25], selected for its superior path quality. We implemented a LoS check that uses 4D collision detection to identify potential conflicts between the spaces reserved by two drones, accounting for both separation volumes in 3-D Euclidean space and their temporal overlap. To manage potential conflicts effectively, we also applied a priority-based resolution strategy.

The problem of modelling in physics normally implies finding a set of suitable instruments to deliver an acceptable level of precision. Having this in mind, we compared the results of the flight dynamics model of rotary-wing UAVs with a super simplified one where the trajectory is calculated based on the maximum velocity of the aircraft. If the deviation is not significant, then the problem of UAV 4-D trajectory planning in U-space could be solved without complex physical modelling during flight. Instead, pre-flight calculations could be done to find the maximum velocity of the UAV for the current case like aircraft weight, air temperature, air pressure, specific cargo transportation, and wind impact.

In this paper, we do not address wind impact, as it is one of the multiple essential constraints in the UAV autonomous guidance [2]. Instead, we focus our study on the fusion of our physical model with the theta-star-based algorithm to test LoS-free flight in our simulation.

The problem of 4-D trajectory planning is a complex multi-factor task with various interconnected constraints [2]. We acknowledge the complexity, and the essentiality of splitting the 4-D trajectory planning problem into the sub-issues. Specifically in this paper, we raise the following research

http://www.imavs.org/

questions:

- How to plan 4-D trajectories for the uncrewed air traffic in order to minimise flight time for each consecutive UAV?
- What is the difference between 4-D trajectory planning based on the Gen4jectory physical model and a model that considers only the maximum velocity of the UAVs?

4 ALGORITHM DESIGN AND METHODOLOGY

To address the research questions, we invented an experimental setup based on the Gen4jectory algorithm [26] (Figure 2). The algorithm predefined:

- The 3-D map used in our experiment has dimensions of 350x350x150 metres, representing the airspace in which the UAVs operate. This airspace is defined as a 3-D rectangle, where the length and width span 350 metres each, and the height extends up to 150 metres above ground level, creating a bounded environment for path planning and navigation tasks.
- We created a lattice graph starting with a 3-D grid of points, spaced evenly at intervals of 25 metres. Each point on the grid became a node in the graph, and we connected these nodes to their neighbouring points with edges, forming a lattice structure.
- The start and goal positions of the UAVs were randomly generated and sampled from a normal distribution within the map's bounds. This allowed us to capture a broad spectrum of potential conflict scenarios, enhancing the robustness of our testing and analysis.
- Separation volumes are defined as 4D-directed rectangles (separation volumes) between two points. Each 4-D rectangle (separation volume) has a length of 30 metres, with the other two dimensions being 10x10 metres.

In the next stage, the simulation randomly generates departure and arrival waypoints for all UAVs involved. Also, it generates random values of aircraft performance for rotary-wing UAVs in the realistic range. Specifically, aircraft mass ranged from 2 kg to 500 kg, which covers a large variety of the U-space airspace participants including flying taxis. The drag coefficient is ranged from 0.6 to 1.2. We chose the range of cross-sectional areas in the range from 0.5 to 4 metres. Additionally we made cross-sectional areas linearly dependent on aircraft mass: The heavier the UAV, the greater its cross-sectional area. Similarly, maximum thrust depends on aircraft mass, ranging from 20% to 100% reserve over aircraft weight. The standard atmosphere is chosen as a typical baseline in aviation research. However, this variable can be adjusted to any value to reflect the real-life temperature and atmospheric pressure impact on flight dynamics.

Next, the algorithm sequentially calculates 4-D trajectories for each UAV. In the case of the LoS threat in a concrete place with concrete time, an alternative trajectory is being calculated. This process continues until a safe near-optimal trajectory with minimised flight time is found. We state "near-optimal" because map representation with predefined waypoints corresponded with deviations from optimality. The greater resolution gives the more optimal solution. However, it also implies a greater time complexity as the theta-star algorithm requires calculating more potential options to navigate. The optimal balance should be found as a compromise based on operational needs. Similarly, the greater the number of UAVs flying nearby, the more complicated check of LoS is required.

In the next stage, a supplementary algorithm makes a simplified calculation of flight time required to fly from one waypoint to another based on UAV maximum level flight speed. However, to make the maximum speed known, we do precalculations based on aircraft performance data and current air density. We cannot rely on the maximum speed given in the flight manual since UAVs can carry extra cargo which affects aerodynamic drag, the temperature can be far from the standard atmosphere, etc.

In the fifth stage, the simulation runs multiple flight experiments while checking: LoS event, flight time for reaching the destination point based on our main physical model in comparison with the super simplified one. Finally, the data is recorded and stored in a table which we use for analysis.

The methodology of this study includes a basic literature review, coding of the experimental setup, performing an analysis of the experiment (simulation) repeated 1,000 times, with random realistic flight characteristics for every UAV, a map 350x350x150 m, random initial positions and destination points of the air space participants, and 5 skyscrapers with random locations. The number of UAVs in flight is 25.

4.1 Physical model

4.1.1 Notation

a - acceleration, m/s^2.
C_D - drag coefficient.
$Chld$ – child waypoint in NEU coordinate system.
dp – dot product of $\boldsymbol{GP\_Par}$.
d_{plnd} - distance planned.
g – acceleration due to gravity, m/s2.
GP – grandparent waypoint in North East Up (NEU) coordinate system.
gp – coordinate of the grandparent waypoint in NEU.
m - UAV mass, kg.
$magGP_Par$ – magnitude of $\boldsymbol{GP\_Par}$.
$magPar\_Chld$ – magnitude of $\boldsymbol{Par\_Chld}$.
M_{end} - a point of the manoeuvre end.
M_{start} - a point of the manoeuvre start.

http://www.imavs.org/

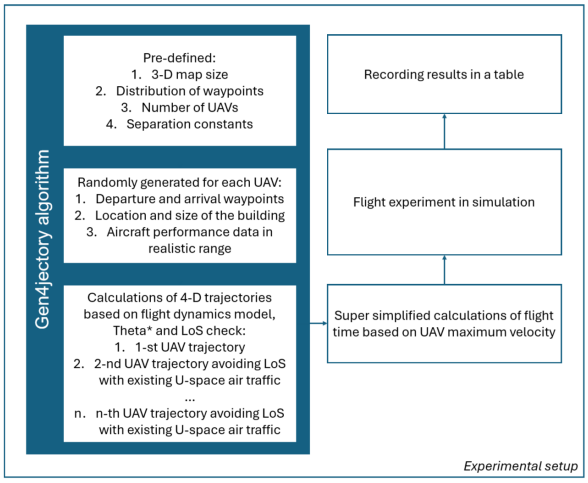


Figure 2: Process Diagram of the experimental setup

Par – parent waypoint in NEU coordinate system.
$prnt$ - coordinate of the parent waypoint in NEU.
S - cross sectional area.
t - flight time, s.
t_{new} - updated time, s.
v - final calculated velocity for a direct section of the flight path, m/s.
v_0 - initial velocity, m/s.
Δ - difference between a vertical component of thrust and UAV weight.
γ - flight path angle, degrees.
$\gamma_{\theta_{lim}}$ - flight path angle for flight with angle θ_{lim} and max thrust, degrees.
ε - angle of the flight path change.
θ - UAV tilt angle, degrees.
θ_{lim} - UAV tilt limit, degrees.
θ_{mh} - UAV tilt for flight in horizon with max thrust, degrees.
ρ - air density, kg/m^3.
$\boldsymbol{F}_{mh}$ - vector of maximum force in horizontal flight without drag, N.
$\boldsymbol{F}_{net}$ - vector of maximum net force, N.
$\boldsymbol{F}_{net_{mh}}$ - vector of maximum net force in horizontal flight, N.
$\boldsymbol{F}_t$ - vector of maximum thrust, N.
$\boldsymbol{F}_{t_{mh}}$ - vector of thrust in horizontal flight, N.
$\boldsymbol{F}_{t_{min}}$ - vector of minimum thrust for vertical descent, N.
$\boldsymbol{F}_{t_{\theta lim}}$ - vector of thrust with θ_{lim} angle, N.
$\boldsymbol{F}_{tr}$ - net force without drag impact, N.
$\boldsymbol{F}_{tr_{hc1}}$ - horizontal component of the net force without drag impact before manoeuvre, N.
$\boldsymbol{F}_{tr_{hc2}}$ - horizontal component of the net force without drag impact during manoeuvre, N.
$\boldsymbol{F}_x$- horizontal component of thrust vector, N.
$\boldsymbol{F}_z$ - vertical component of a corresponded vector, N.
$\boldsymbol{GP\_Par}$ - vector from waypoint GP to waypoint Par.
$\boldsymbol{Par\_Chld}$ - vector from waypoint Par to waypoint $Chld$.

Subscript N/E/U stands for North/East/Up coordinate accordingly. We name some values with "$_{red}$" in case the reduced thrust is discussed.

All symbols representing force vectors in this paper we denote by bold letters. The corresponding magnitudes, when referenced, we denote by italic letters without bold formatting.

4.1.2 Time-dependent UAV motion modelling

In the paper we consider the next conditions:
- NEU Body-fixed coordinate system.
- NEU origin at sea level at some point.
- Mass-point model of UAV.
- Earth curvature is ignored.
- Wind is zero.

To plan a 4-D trajectory safely, it is essential to reserve a certain volume of airspace at a certain time to avoid the risk of LoS. Vertical, horizontal, and longitudinal separation create a 4-D rectangle, whose sides are normally defined by national regulation. However, to reduce computation, the 4-D rectangle can be simplified as a set of safety bubbles placed on the aircraft's current, most recent past, and nearest future positions. If there is no presence of two aircraft in the same safety area, then the risk of potential collision is significantly mitigated. To make sure that there is no potential presence of two aircraft in the same volume of airspace, it is required to calculate where and when the aircraft will be. For that, it is essential to calculate flight time, initial and final velocity, and to address the inertia issue during manoeuvres. We created the code [26] that resolves this issue.

Following a common practice in aviation research, we assume that the flight occurs under standard atmospheric conditions with no wind. To calculate the flight time required to cover a distance between two waypoints, the physical model has three layers: inputs, calculations, and outputs. We assume that inputs are all given properties of the drone. Among them: mass (m), cross-sectional area (S) and drag coefficient (C_D), the maximum thrust (F_t), the maximal tilt angle of the UAV (θ_{lim}), initial velocity (v_0), air density (ρ), minimum percent of thrust for vertical descending, and delta time for Euler method calculations. The coordinates of the waypoints in the NEU reference system are also known.

Based on the inputs provided, the algorithm transforms Parent (departure) and Child (arrival) waypoints to a vertical plane where the Parent waypoint is placed in the origin. We switch to a body-fixed reference system where the z-axis is pointed up and the x-axis is directed to the Child (next) waypoint. In this case, it is essential to find the coordinates of the Parent and Child waypoint for each straight segment of the trajectory. Once the UAV has reached the area of the

http://www.imavs.org/

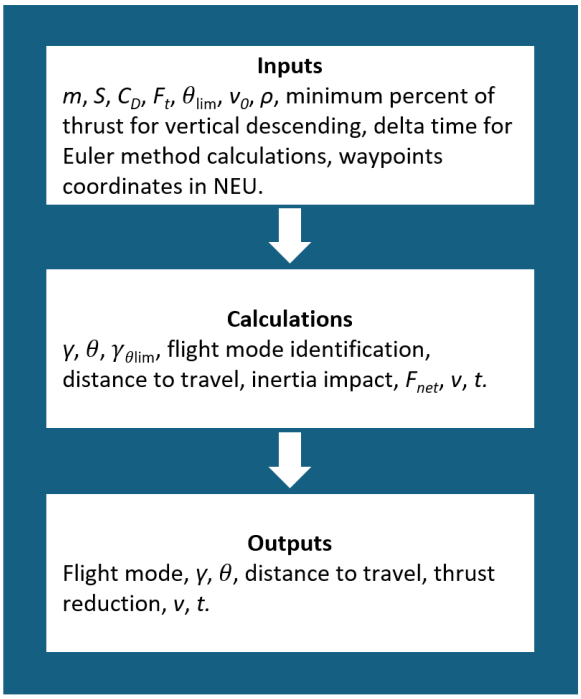


Figure 3: The high-level scheme of physical model calculations

arrival point, the algorithm switches through the loop to the next segment, and the Child point is now classified as the Parent point. It happens until the final destination waypoint has been reached.

There are a few steps to perform a transformation to the new coordinate system. Firstly, it is essential to select the departure and arrival waypoints that correspond to the segment required. Then we define two vectors corresponding to the departure and arrival waypoints.

$$\mathbf{W}_{\text{chld}} = (N_{\text{chld}},\, E_{\text{chld}},\, U_{\text{chld}}) \tag{1}$$

$$\mathbf{W}_{\text{par}} = (N_{\text{par}},\, E_{\text{par}},\, U_{\text{par}}) \tag{2}$$

A relative position vector can be found:

$$\mathbf{W}'_{\text{chld}} = \mathbf{W}_{\text{par}} - \mathbf{W}_{\text{chld}} \tag{3}$$

We ignore the vertical component and normalise the horizontal component of $\mathbf{W}_{\text{chld}}$', introducing new variables h_s, x_{axis}, Z_{axis}.

$$h_s = \begin{pmatrix} W'_{\text{chldN}} \\ W'_{\text{chldE}} \\ 0 \end{pmatrix} \tag{4}$$

$$x_{\text{axis}} = \frac{h_s}{\|h_s\|} \tag{5}$$

$$Z_{\text{axis}} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{6}$$

Arrival waypoint x-coordinate and z-coordinate in body-fixed coordinate system:

$$w_x = (w_{\text{par}} - w_{\text{chld}}) \cdot x_{\text{axis}} \tag{7}$$

$$w_z = (w_{\text{par}} - w_{\text{chld}}) \cdot z_{\text{axis}} \tag{8}$$

Now the arrival point is transformed to the origin, and accordingly, the destination point has x and z coordinates.

The basic aerodynamic forces and maximum tilt angle θ have a significant impact on finding a flight mode that allows minimising flight time. With a high thrust-to-weight ratio, the aircraft will be ascending with angle $\gamma_{\theta_{lim}}$ if it has the maximum tilt angle with the maximum thrust.

However, if we significantly increase the weight of UAV, it could lead to a low thrust-to-weight ratio. In this case, the aircraft will be descending with angle $\gamma_{\theta_{lim}}$ at the maximum tilt angle with the maximum thrust. We mark flight modes for the high thrust-to-weight ratio with the capital letter ´A´ and for the low thrust-to-weight ratio with the capital letter ´B´.

Until the UAV has reached $\gamma_{\theta_{lim}}$ it shall fly at maximum thrust to minimise flight time - modes IA, IB, and IIB (Figures 4 and 5). Exceptions are possible for the aggressive style flight with multiple changes of flight directions. For U-space traffic, we expect mainly direct trajectories with limited manoeuvring.

Once $\gamma_{\theta_{lim}}$ is reached, the descent with γ in range (- π/2, $\gamma_{\theta_{lim}}$) is only possible if thrust is decreased - flight modes IIA, IIIA, and IIIB (Figures 4 and 5).

Table 1 identifies the significant differences between the flight modes. For example, vertical ascend and descent imply zero tilt angle. However, the maximum thrust is required for the fastest ascent. In order to maintain aircraft orientation, the minimum thrust can be applied for the vertical descending. Flight modes IA, IB, and IIB imply maximum thrust; however, UAV tilt angle requires calculations. Flight modes IIA, IIIA, and IIIB operate at the maximum tilt angle; however, the level of thrust in Newtons must be calculated.

Flight mode	**Angle γ**	**Thrust**	**Angle θ**
Vertical ascent	$\pi/2$	Max	0
IA	$[\gamma_{\theta_{\text{lim}}}, \pi/2)$	Max	To find
IB	$[0, \pi/2)$	Max	To find
IIB	$[\gamma_{\theta_{\text{lim}}}, 0)$	Max	To find
IIA	$[0, \gamma_{\theta_{\text{lim}}})$	To find	θ_{lim}
IIIA	$(-\pi/2, 0)$	To find	θ_{lim}
IIIB	$(-\pi/2, \gamma_{\theta_{\text{lim}}})$	To find	θ_{lim}
Vertical descent	$-\pi/2$	As input	0

Table 1: Flight modes

http://www.imavs.org/

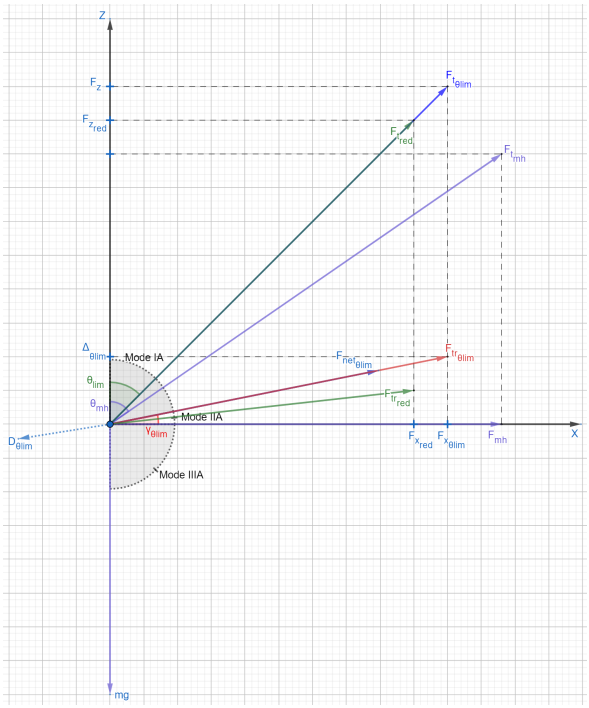


Figure 4: Flight modes A. For rotary-wing UAV with a high thrust-to-weight ratio ($\gamma_{\theta_{lim}} \geq 0$)

Travel distance is essential to find the flight time required and the velocity at the end of the segment.

$$d_{\text{plnd}} = \sqrt{x^2 + z^2} \tag{9}$$

We propose a new variable b to make calculations look simple.

$$b = \frac{1}{2} \cdot \rho \cdot S \cdot C_D \cdot V^2 \tag{10}$$

As waypoint coordinates are given, it is possible to find flight path angle γ:
If x $>$ 0, then γ = $\arctan(z/x)$.
If x = 0 and z $>$ 0, then γ=π/2.
If x = 0 and z $<$ 0, then γ=-π/2.
If x = 0 and z = 0, then the waypoint is at the origin.

No need to change UAV position. Now γ is known, thus θ can be found which is required for the flight modes IA, IB, and IIB according to table 1. As Δ is a vertical component of $\boldsymbol{F}_{tr}$ and $\boldsymbol{F}_x$ is a horizontal component of $\boldsymbol{F}_{tr}$, then:

$$F_{tr} = \sqrt{\Delta^2 + F_x^2} \tag{11}$$

$$\sin\gamma = \frac{\Delta}{F_{tr}} \tag{12}$$

$$\Delta = \sin\gamma \cdot \sqrt{\Delta^2 + F_x^2} \tag{13}$$

The magnitude of the horizontal force is found.

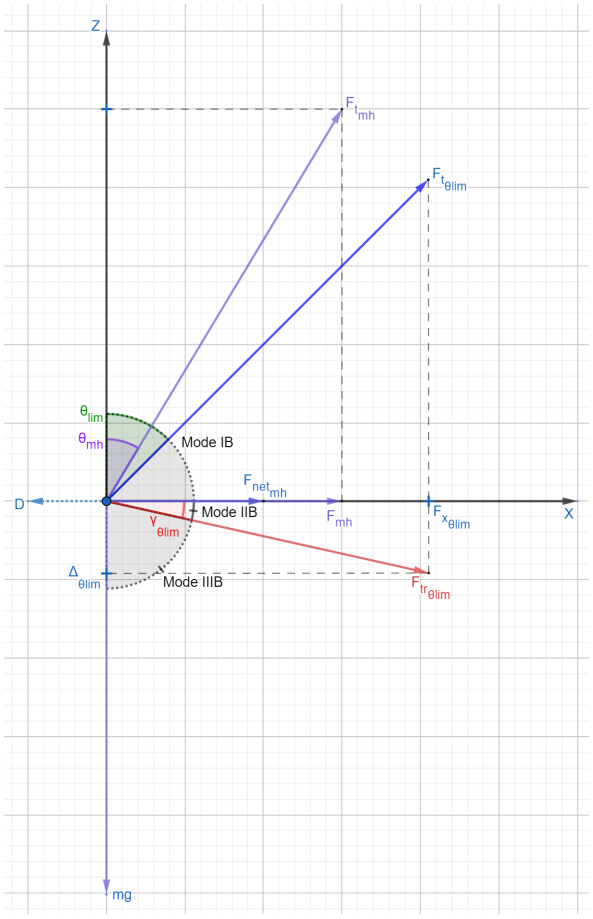


Figure 5: Flight modes B. For rotary-wing UAV with a low thrust-to-weight ratio ($\gamma_{\theta_{lim}} < 0$)

$$F_x = \sqrt{F_t^2 - (\Delta + m \cdot g)^2} \tag{14}$$

$$\Delta = \sin\gamma \cdot \sqrt{\Delta^2 + F_t^2 - (\Delta + m \cdot g)^2} \tag{15}$$

$$\Delta^2 - \sin^2\gamma \cdot \left(\Delta^2 + F_t^2 - \Delta^2 - 2 \cdot \Delta \cdot m \cdot g - (m \cdot g)^2\right) = 0 \tag{16}$$

$$\Delta^2 + 2 \cdot \sin^2\gamma \cdot m \cdot g \cdot \Delta - \sin^2\gamma \cdot F_t^2 + \sin^2\gamma \cdot (m \cdot g)^2 = 0 \tag{17}$$

To make the equation look simple, we enter p and c.

$$p = 2 \cdot \sin^2\gamma \cdot m \cdot g \tag{18}$$

$$c = -\sin^2\gamma \cdot F_t^2 + \sin^2\gamma \cdot (m \cdot g)^2 \tag{19}$$

$$\Delta^2 + p \cdot \Delta + c = 0 \tag{20}$$

http://www.imavs.org/

$$D = p^2 - 4c \tag{21}$$

$$\Delta_{1,2} = \frac{-p \pm \sqrt{p^2 - 4c}}{2} \tag{22}$$

In the case of flight mode IA and IB $\Delta = \Delta_1$. For IIB $\Delta = \Delta_2$.

$$\cos\theta = \frac{\Delta + m \cdot g}{F_t} \tag{23}$$

$$\theta = \arccos\left(\frac{\Delta + m \cdot g}{F_t}\right) \tag{24}$$

As θ is calculated, it is possible to find the horizontal component of the thrust vector for the modes IA, IB, and IIB:

$$F_x = \sin\theta \cdot F_t \tag{25}$$

And F_{tr} can be found with an equation (11).

For the modes IIA, IIIA, and IIIB, the UAV flies at θ_{lim} and the reduced thrust shall be calculated. In this case, Δ is different.

$$\sin(\theta_{\text{lim}}) = \frac{F_{x\_red}}{F_{t\_red}} \tag{26}$$

$$\tan(\theta_{\text{lim}}) = \frac{F_{x\_red}}{m \cdot g + \Delta_{\text{red}}} \tag{27}$$

$$F_{x\_red} = \tan(\theta_{\text{lim}}) \cdot (m \cdot g + \Delta_{\text{red}}) \tag{28}$$

$$\tan\gamma = \frac{\Delta_{\text{red}}}{F_{x\_red}} \tag{29}$$

$$F_{x\_red} = \frac{\Delta_{\text{red}}}{\tan\gamma} \tag{30}$$

$$\tan(\theta_{\text{lim}}) \cdot (m \cdot g + \Delta_{\text{red}}) = \frac{\Delta_{\text{red}}}{\tan\gamma} \tag{31}$$

$$\Delta_{\text{red}} - \tan\gamma \cdot \tan\theta_{\text{lim}} \cdot m \cdot g - \tan\gamma \cdot \tan\theta_{\text{lim}} \cdot \Delta_{\text{red}} = 0 \tag{32}$$

$$\Delta_{\text{red}} = \frac{\tan\gamma \cdot \tan\theta_{\text{lim}} \cdot m \cdot g}{1 - \tan\gamma \cdot \tan\theta_{\text{lim}}} \tag{33}$$

F_{x_red} is known, see equation (28).

$$F_{tr_{\text{red}}} = \sqrt{\Delta_{\text{red}}^2 + F_{x\_red}^2} \tag{34}$$

Table 2 summarises the similarities and differences in the calculations in finding the vector of net force without drag impact. For example, vertical ascend and descent imply known thrust. Minimum thrust is essential for vertical descent. Flight modes IA, IB, and IIB have commonality in finding F_x and F_{tr}, but IIB differs in finding Δ and θ. Flight modes IIA, IIIA, and IIIB have the same line of calculations.

Flight mode	Similarities	Differences
Vertical ascend	F_t is given	
IA	*(11), (25)*	$\Delta = \Delta_1, \theta = \theta_1$
IB		
IIB		$\Delta = \Delta_2, \theta = \theta_2$
IIA	*(28), (33), (34)*	
IIIA		
IIIB		
Vertical descend	Given F_t and F_{t_red}	

Table 2: Comparison of flight modes

F_{tr} has been calculated for each mode, and the net force can be found via this equation:

$$F_{\text{net}} = F_{\text{tr}} - \frac{1}{2} \cdot \rho \cdot S \cdot C_D \cdot V^2 \tag{35}$$

The acceleration can be found through Newton's law.

$$a = \frac{F_{\text{net}}}{m} \tag{36}$$

Finding flight time between waypoints and final velocity, position update, and time update can be calculated via Euler's method.

$$v_{t+\Delta t} = v_t + a_t \cdot \Delta t \tag{37}$$

$$x_{t+\Delta t} = x_t + v_t \cdot \Delta t \tag{38}$$

$$t_{\text{new}} = t + \Delta t \tag{39}$$

Now, the travelling time between two waypoints can be calculated.

The calculations we did so far considered only two waypoints - Parent and Child. That is not an issue for the first direct segment of the 4-D trajectory because we assume that the initial velocity is known, and it is zero (the UAV takes off from a vertiport). However, in case of changing flight path direction, the inertia gets the higher impact the greater the angle of manoeuvre. Since we intend to plan a 4-D trajectory based on limited information about the UAV, a simplified model of inertia is required. In this light, we have to consider three waypoints - Grandparent (GP), Parent (Par), and Child ($Chld$). If we put a plane on the three waypoints, then it is possible to draw the manoeuvre in 2-D space which is simple for understanding. One of the cases is a horizontal manoeuvre in the vicinity of the Parent waypoint (Figure 6). Since rotary-wing UAVs have a small turning radius in comparison to separation requirements, it is possible to neglect the more precise calculations of the 4-D trajectory and accept the corresponding deviations.

http://www.imavs.org/

http://www.imavs.org/

Figure 6 gives an example of a 90-degree manoeuvre in a plane based on the initial three waypoints. The model implies minor deviations due to simplification: The UAV travels with a cutting angle and does not reach the Parent point precisely. The strong side of this approach is the simplicity of addressing inertia with having limited data on the rotary-wing UAV.

To simplify the calculation, we assume that a UAV approaching the Parent's waypoint vicinity can turn on the angle ε in advance (Figure 6) - in the point M_{start}. Due to inertia, the UAV will continue moving in its current state of motion, but aerodynamic drag will gradually decelerate it, potentially causing the UAV to move towards the Parent-Child segment. It is not necessary to find the precise position of M_{start}; this point is essential just for understanding how the issue of inertia can be addressed in a simplified way. However, angle ε is essential to calculate as it has a high impact on the initial velocity for the Parent-Child segment. For example, for very small ε, the final velocity for the Grandparent-Parent segment will be almost equal to the initial velocity of the Parent-Child. For $\varepsilon = \pi$, the initial velocity of the Parent-Child segment will be equal to the negative meaning of the final velocity for the Grandparent-Parent segment.

Since v is the final calculated velocity for a direct section of the flight path, and v_0 is the initial velocity, the next equation can be used as a simplification of the manoeuvre for any angle ε.

$$v_0 = \cos\varepsilon \cdot v \tag{40}$$

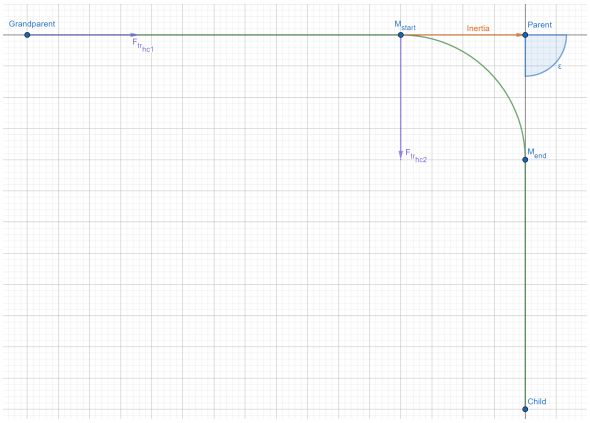


Figure 6: Manoeuvre at 90 degrees on the constant altitude. View from above

To find the angle ε, some calculations are required. The coordinates of three waypoints in NEU are known.

$$GP = \left[gp_N, gp_E, gp_U\right] \tag{41}$$

$$Par = \left[prnt_N, prnt_E, prnt_U\right] \tag{42}$$

$$Chld = \left[chld_N, chld_E, chld_U\right] \tag{43}$$

To compute vectors from point:

$$\boldsymbol{GP\_Par} = (prnt_N - gp_N, prnt_E - gp_E, prnt_U - gp_U) \tag{44}$$

$$\boldsymbol{Par\_Chld} = (chld_N - prnt_N, chld_E - prnt_E, chld_U - prnt_U) \tag{45}$$

To find dot product:

$$\begin{aligned} \text{dp} = {} & \text{GP\_Par}_N \cdot \text{Par\_Chld}_N + \text{GP\_Par}_E \cdot \text{Par\_Chld}_E \\ & + \text{GP\_Par}_U \cdot \text{Par\_Chld}_U \end{aligned} \tag{46}$$

To find magnitudes:

$$\text{mag}_{GP\_Par} = \sqrt{GP\_Par_N^2 + GP\_Par_E^2 + GP\_Par_U^2} \tag{47}$$

$$\text{mag}_{Par\_Chld} = \sqrt{Par\_Chld_N^2 + Par\_Chld_E^2 + Par\_Chld_U^2} \tag{48}$$

To find angle ε:

$$\cos\varepsilon = \frac{dp}{\text{mag}_{\text{GP\_Par}} \cdot \text{mag}_{\text{Par\_Chld}}} \tag{49}$$

$$\varepsilon = \arccos(\cos\varepsilon) \tag{50}$$

4.2 Map representation

In our study, the 3-D environment is modelled as large, bounded airspace with dimensions of 350x350x150 metres. This defined space simulates a real-world environment where UAVs need to navigate, such as in urban or controlled airspace. To effectively manage pathfinding within this environment, we constructed a grid-graph representation of the map. The grid is created by placing into the airspace the evenly spaced points, with intervals set to one-fifth of the map's total length. Each point on this grid becomes a node in our graph, and these nodes are interconnected by edges to their nearest neighbours, forming a lattice structure.

The start and goal positions for the UAVs are not just placed arbitrarily; they are generated using a normal probability distribution over the original continuous map. After the grid-graph map is built, these start and goal positions are incorporated into the graph as new nodes. We then create edges connecting these nodes to their nearest neighbours in the grid, integrating them seamlessly into the existing structure.

This method allows us to simulate a wide range of possible flight paths and conflict scenarios, providing a robust foundation for testing our pathfinding and collision avoidance algorithms. By combining a structured grid with randomly generated positions, we ensure that the UAVs are challenged to navigate efficiently and safely through a realistic 3-D environment.

4.3 Pathfinding technique

For efficient pathfinding in the 3-D environment, we implemented the theta-star algorithm, an advanced variation of the A-star algorithm tailored to handle the complexities of three-dimensional space. Unlike A-star, which strictly follows the grid's structure and generates paths constrained by the grid's edges, theta-star introduces the concept of line-of-sight between nodes. This capability allows theta-star to bypass unnecessary intermediate nodes by directly connecting a node to any visible successor, effectively "cutting corners" to produce shorter and smoother paths.

The main advantage of theta-star lies in its ability to reduce path length and optimise the path's smoothness, which is particularly important in 3-D environments where UAVs must navigate around static and dynamic obstacles. The algorithm operates by relaxing the parent-child relationship between nodes seen in A-star, allowing a node to be directly connected to its grandparent if the direct path is clear, thus reducing the number of turns and deviations in the trajectory. This leads to more efficient paths that minimise the UAVs' travel time and energy consumption.

In this application, the theta-star algorithm is integrated with a flight dynamics model, which accounts for the most essential physical constraints and manoeuvrability of the UAVs. This ensures that the paths generated are not only near-optimal in terms of distance but also feasible given the UAVs' dynamic capabilities. By combining theta-star with our flight dynamics model, we generate paths that are both computationally efficient and practical for real-world navigation, especially in complex environments with obstacles such as buildings or other UAVs.

4.4 LoS check

Each UAV in our system has a separation volume, modelled as a 4-D rectangle that spans both 3-D space and time. This separation volume is directed between two points the UAV will visit, and it consists of three parts: a middle segment, where the UAV is actively travelling, which is 10 metres long, and forward and back segments of the same length, this makes the total of 30 metres in length. These additional segments provide extra safety by extending the protected area around the UAV, reducing the risk of a LoS with other UAVs.

During the trajectory planning phase, the theta-star algorithm evaluates potential paths by examining the edges between different nodes. The theta-star checks if the edge between two nodes intersects any already reserved separation volumes by other UAVs. If the edge does not intersect a reserved volume, the transition to the next node in the theta-star algorithm is considered safe. If the edge is part of the optimal path, the associated separation volume is then reserved to prevent other UAVs from planning trajectories that would conflict with it.

Collision detection between separation volumes is managed by placing collision spheres along the long axis of the directed 4-D rectangle. These spheres have a radius equal to the separation volume width which is 10 metres and are spaced 1 metres apart. If any of these collision spheres from one separation volume intersects with those from another, and the separation volumes overlap in time, a LoS violation is detected. When this happens, the next node in the path is ignored, indicating an obstacle along the UAV's route.

5 SIMULATION RESULTS

We planned and performed 25,000 flights in 1,000 simulations for 25 rotary-wing UAVs flying in the shared airspace 350x350x150 metres with 5 skyscrapers in standard atmosphere (Figure 1). UAV flight characteristics and all positions were random for each case. For example, the mass of drones ranged from 2 to 500 kg. As a result, the Gen4jectory algorithm successfully planned the uncrewed air traffic for all cases. The LoS case was not registered for any flight which confirms the capability of the algorithm to plan collision-free 4-D trajectories based on the unique flight characteristics of the rotary-wing aircraft.

Simultaneously, the experimental setup calculated the maximum airspeed for every UAV based on its unique aircraft performance data - drag coefficient, maximum thrust, cross-sectional area, and mass. The interesting finding was that a physical model of the Gen4jectory algorithm gave flight time calculations very close to the calculation considering only maximum airspeed. The deviation was about 0-2%.

The calculations to find the maximum speed in level flight can be done before a flight. However, the calculated maximum speed can significantly differ from the value in the flight manual, for example, due to changes in air temperature, and/or aircraft mass. Also, drag coefficient and cross-sectional area can differ significantly due to the outboard cargo.

6 CONCLUSION

Based on the experimental results, we conclude that the Gen4jectory algorithm can plan 4-D trajectories for rotary-wing UAVs, even in congested urban environments, within the assumptions we made. The algorithm guarantees that all trajectories will be planned without LoS in a near-optimal way considering the unique aircraft performance data of every drone.

The algorithm realised the idea of equal access to the airspace. It implies that the sooner a UAV requests a 4-D trajectory, the more optimal trajectory the drone receives from the algorithm. Nevertheless, we expect further research on

http://www.imavs.org/

the holistic optimisation of uncrewed air traffic as an alternative to our approach.

Another path for further research can be an optimisation of the algorithm using precalculated maximum speed instead of numerical methods for flight time calculations. As we know from the experiment, the difference is about a few percent. However, we can expect significant optimisation in computation time.

REFERENCES

[1] McKinsey & Company. Study on the societal acceptance of Urban Air Mobility in Europe. *European Union Aviation Safety Agency*, May 2021. Accessed: Jul. 06, 2022.

[2] I. Panov and A. Ul Haq. A Critical Review of Information Provision for U-Space Traffic Autonomous Guidance. *Aerospace*, 11(6), 2024.

[3] Single European Sky ATM Research 3 Joint Undertaking (EU body or agency). *European drones outlook study: unlocking the value for Europe*. Publications Office of the European Union, 2017. Accessed: Aug. 27, 2024.

[4] A. Islami, S. Chaimatanan, and D. Delahaye. *Large-Scale 4D Trajectory Planning*, pages 27–47. Springer Japan, Tokyo, 2017.

[5] M. Sun, K. Rand, and C. Fleming. 4 Dimensional waypoint generation for conflict-free trajectory based operation. *Aerospace Science and Technology*, 88:350–361, May 2019.

[6] E. Gallo, J. Lopez-Leones, M. A. Vilaplana, F. A. Navarro, and A. Nuic. Trajectory computation Infrastructure based on BADA Aircraft Performance Model. In *2007 IEEE/AIAA 26th Digital Avionics Systems Conference*, pages 1.C.4–1–1.C.4–13, Dallas, TX, USA, 2007. Institute of Electrical and Electronics Engineers.

[7] S. Tang, K. Sreenath, and V. Kumar. *Multi-robot Trajectory Generation for an Aerial Payload Transport System*, pages 1055–1071. Springer International Publishing, Cham, 2020.

[8] Q. He, R. Wang, Z. Wang, and Y. Wang. A Dynamic Trajectory Planning Algorithm for Urban Ultra-low Altitude UAVs Based on a Fusion Heuristic Algorithm. In *2023 6th International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pages 148–153, May 2023.

[9] Y. Zuqiang, F. Zhou, and L. Ping. A Bio-inspired Collision-free 4D Trajectory Generation Method for Unmanned Aerial Vehicles Based on Tau Theory. In *2015 34th Chinese Control Conference (CCC)*, pages 6961–6968, Jul. 2015.

[10] S. Aggarwal and N. Kumar. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. *Computer Communications*, 149:270–299, Jan. 2020.

[11] C. A. P. Neto, G. de C. Bertoli, and O. Saotome. A-star path planning simulation for UAS Traffic Management (UTM) application, Jul. 2021.

[12] X. Lin, C. Wang, K. Wang, M. Li, and X. Yu. Trajectory planning for unmanned aerial vehicles in complicated urban environments: A control network approach. *Transportation Research Part C: Emerging Technologies*, 128:103120, Jul. 2021.

[13] F. Ho et al. Decentralized Multi-Agent Path Finding for UAV Traffic Management. *IEEE Transactions on Intelligent Transportation Systems*, 23(2):997–1008, Feb. 2022.

[14] K. Daniel, A. Nash, S. Koenig, and A. Felner. Theta*: Any-Angle Path Planning on Grids. *jair*, 39:533–579, Oct. 2010.

[15] A. Nash, S. Koenig, and C. Tovey. Lazy Theta*: Any-Angle Path Planning and Path Length Analysis in 3D. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, Jul. 2010.

[16] M. Yuan, T. Zhou, and M. Chen. Improved lazy theta* algorithm based on octree map for path planning of UAV. *Defence Technology*, 23:8–18, May 2023.

[17] D. Foead, A. Ghifari, M. B. Kusuma, N. Hanafiah, and E. Gunawan. A Systematic Literature Review of A* Pathfinding. *Procedia Computer Science*, 179:507–514, Jan. 2021.

[18] J. R. Sánchez-Ibáñez, C. J. Pérez del Pulgar, and A. García-Cerezo. Path Planning for Autonomous Mobile Robots: A Review. *Sensors*, 21(23), Jan. 2021.

[19] M. Faria, A. S. Ferreira, H. Pérez-Leon, I. Maza, and A. Viguria. Autonomous 3D Exploration of Large Structures Using an UAV Equipped with a 2D LIDAR. *Sensors*, 19(22), Jan. 2019.

[20] M. Xue and M. Wei. Small UAV Flight Planning in Urban Environments. In *AIAA AVIATION 2020 FORUM*. American Institute of Aeronautics and Astronautics, 2020.

[21] R. Rey, J. A. Cobano, L. Merino, and F. Caballero. Adaptation of Lazy-Theta* for UAS 3D path planning considering safety costs. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 387–393, Jun. 2021.

http://www.imavs.org/

[22] X. He, F. He, L. Li, L. Zhang, and G. Xiao. A Route Network Planning Method for Urban Air Delivery. *Transportation Research Part E: Logistics and Transportation Review*, 166:102872, Oct. 2022.

[23] M. Faria, R. Marín, M. Popović, I. Maza, and A. Viguria. Efficient Lazy Theta* Path Planning over a Sparse Grid to Explore Large 3D Volumes with a Multirotor UAV. *Sensors*, 19(1), Jan. 2019.

[24] M.-H. Do, C.-E. Lin, and Y.-C. Lai. Validation of the Flight Dynamics Engine of the X-Plane Simulator in Comparison with the Real Flight Data of the Quadrotor UAV Using CIFER. *Drones*, 7(9), Sep. 2023.

[25] D. Mandloi, R. Arya, and A. K. Verma. Unmanned aerial vehicle path planning based on A* algorithm and its variants in 3d environment. *International Journal of System Assurance Engineering and Management*, 12(5):990–1000, Oct. 2021.

[26] M. Boumediene and I. Panov. Gen4jectory algorithm and experimental setup, Sep. 2024. Python, https://gitlab.com/uaswork/gen4jectory.

http://www.imavs.org/

Control of the DarkO Tail-Sitter Drone through an LMI-Based Static Output Feedback Design

Armand-Ioan Curpanaru*[1,2] and Fabrice Demourant†[1] and Florian Sansou‡[3]
[1]ONERA/DTIS, Toulouse, France, [2]ISAE-SUPAERO, Toulouse, France
[3] ENAC, Université de Toulouse, F-31400 Toulouse, France

http://www.imavs.org/

ABSTRACT

This paper deals with the control of a convertible drone through the synthesis of Static Output Feedback (SOF) model-based controllers. To that end, a convex optimization algorithm based on Lyapunov's stability theory and Linear Matrix Inequalities (LMI) available in the literature is employed. To evaluate the algorithm's performance, given that it has never been previously tested on realistic dynamical systems, it is implemented on the experimental model of the DarkO drone. Then, simulations of the closed-loop dynamic of the drone are carried out to assess the performance of the new design. For a preliminary experimental validation, the control architecture is implemented on the real drone system and test flights focused on the hovering phase and reference tracking in terms of position are carried out.

1 INTRODUCTION

1.1 Forewords and problem description

Despite numerous technological advances, today's drones are still limited by their energy consumption and lack of adaptation to complex urban environments. To reduce these weaknesses, a new generation of drones has been developed, namely the convertible drones that offer the capability of taking-off and landing vertically, while having a high aerodynamical efficiency in cruise.

The present article positions itself along a new direction of research aimed at developing control architectures for convertible UAVs (Unmanned Aerial Vehicles). The different designs and control techniques of hybrid and convertible VTOL (Vertical Take-Off and Landing) platforms are presented and discussed in [1]. The decision of employing a particular control scheme is mainly motivated by its performance in stabilizing and controlling the plant's dynamics across multiple phases such as take-off, hovering, transition, horizontal flight and landing.

Research supported in part by ONERA-ISAE-ENAC Research Federation
* armand-ioan.curpanaru@onera.fr, armand-ioan.curpanaru@isae-supaero.fr
† fabrice.demourantr@onera.fr
‡ florian.sansou@recherche.enac.fr

With respect to the controller synthesis process, model-based or model-free/data-driven methodologies are employed. Study [2] presents a detailed comparison between model-based and model-free control. Model-based methods [3, 4, 5] require sophisticated wind-tunnel characterization as well as accurate modeling of propeller-wing interactions, forces and moments of a partially stalled wing and of control surfaces that is highly challenging and time-consuming. As an alternative, model-free control [6, 7] does not need sophisticated models, as it is generally based on sensor measurements for estimating a large part of the dynamic model. As a main drawback, such techniques use test flight data to tune offline the control coefficients. There are also methods like INDI (Incremental Nonlinear Dynamic Inversion) [7] that integrate model-based and sensor-based methods, but are limited by the lack of proof of convergence.

The controller synthesis approach that has been chosen is a model-based one for obtaining SOF (Static Output Feedback) controllers. The majority of controllers implemented for this drone concept are data-driven ones. As a result, designing model-based controllers is a real challenge, having not been addressed exhaustively by the research community. Additionally, considering the intended operation of the DarkO drone in urban and populated areas, it is conceivable that future certification may become necessary. In such a scenario, a model-based control architecture significantly streamlines the certification process.

SOF represents the simplest control law architecture, since only static gains are implemented in the closed loop for generating command signals from measured states. In brief, the implementation is straightforward. Moreover, the fact of explicitly using measurements instead of state variables makes this type of control law extremely attractive from an engineer's point of view.

Beyond the apparent controller simplicity, the stabilizing SOF gain synthesis is a well-known *NP-hard* control problem [8] that cannot be solved by classical synthesis tools [9]. This *NP-hardness* feature comes from the BMI (Bilinear Matrix Inequality) problem formulation. Let us note that any fixed or low-order dynamic controller can be reparametrized as a SOF [10],[11], making this controller structure relatively versatile. More specifically, for a dynamic controller the problem is recast as SOF synthesis by considering the state space representaion of the controller as gain matrices to be optimized. Due to the *NP-hardness* of the SOF problem, it

http://www.imavs.org/

is difficult to implement a systematic method that can either find a solution to the SOF problem or evaluate the feasibility problem. In the literature, the approaches are mainly based on either non-smooth optimization techniques for obtaining a stabilizing controller under structure constraints [12] or on Lyapunov's stability theory [10], where LMIs are iteratively solved, to treat the BMI original problem. The LMI-based approaches are of interest, since the same framework is used for robust control. Furthermore, the methods for designing a SOF controller can be easily extended to the uncertain and robust (R-SOF) case for a polytopic model or multimodel representation.

1.2 Outlines

The convertible drone design, having a low research maturity in the field of Automatic Control, presents a high degree of potential novelty that is exploited in this article. The main contributions are represented by the implementation of a mathematical formulation for model-based controller synthesis on a practical case study. Thus far, this approach has been tested and evaluated only on theoretical cases [13] with dynamic models that did not reflect real-world systems.

The proposed method for controlling the studied convertible drone model is based on synthesising a SOF controller, that is obtained by algorithmically converting the non-convex BMI problem to 3 auxiliary convex problems i) a State Feedback (SF) problem, ii) an Output Injection (OI) problem, and iii) a State Injection (SI) problem. Solving the SF, OI, and SI problems are necessary conditions for the existence of a solution for the SOF problem. Consequently, solving each of the above problems is addressed through a 3-phase algorithm presented in section 3. This algorithm is implemented on DarkO's augmented dynamics which are incorporated into the open loop to enforce a desired shape, akin to the principles of the loop shaping method [14]. This drives to multiple guarantees, namely in performance, with open-loop high gain at low frequency, and in robustness, with open-loop weak gain strategy at high frequency while also in satisfying the reference tracking requirement.

The resulting SOF controllers for the augmented dynamics will be validated through time-domain simulations. Moreover, the majority of research articles that introduce control system solutions for convertible drones are limited to simulation-only validation, whereas in this article, the validation will also be conducted experimentally through real flight tests with an experimental model of the DarkO drone.

The article is organized as follows. The general methodology is introduced in section I. In section II, the non-linear and linear dynamics modeling is presented. Section III, details the deterministic iterative algorithm for SOF design that will be implemented, as well as the plant's augmentation process. In section IV, the results of the controller synthesis are analysed through time-domain as well as experimental simulations, by implementing the designed control laws directly on the real system.

1.3 Notations

$He\{X\} = X + X^T$, $(.)^T$ is the transpose of a matrix while the identity matrix and the null matrix of dimension p read, respectively, I_p and 0_p, $(X)^\circ$ is the full rank matrix s.t. $X^\circ X = 0$, $(X)^\perp$ is the full rank matrix s.t. $XX^\perp = 0$. $\mathcal{B}$ and $\mathcal{I}$ are the body and inertial reference frames, respectively, while x_b, y_b, z_b and x_0, y_0, z_0 are their associated axis systems.

2 MODELING OF DARKO DYNAMICS

The unmanned VTOL platforms are mainly split into tailsitter-, tiltrotor-, and tiltwing-VTOL aircraft. The model that will be used as a study case is the DarkO drone, a micro convertible tail-sitter flying wing, developed at ENAC Toulouse Fig.1. It is fitted with two propellers (shown in **black**) that generate thrust on the longitudinal axis and moments around the yaw axis, together with two control surfaces, namely elevons (shown in blue) that can be operated independently for generating moments around the pitch and roll axis. The dynamics have been modelled in previous research studies [15] and flight tests have been conducted [5] using the INDI method.

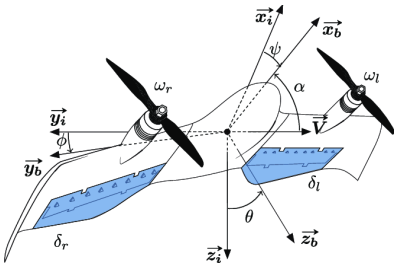


Figure 1: DarkO drone

It has to be stated that the dynamics on the DarkO drone are highly nonlinear. This is due to the aerodynamic forces and moments that vary with the square of the airspeed v^2, as well as couplings between actuators, notably the yawing moment generated by an elevon deflection, that varies with the airspeed of the flow over the control surface which in turn depends on the angular speed of the propeller. Additional nonlinearities emerge from the nonlinear dynamic of the quaternion q as well as products between the states (for example $w_{bx}w_{by}$, $w_{by}w_{bz}$) or higher order states (such as w_{bx}^2). There are even nonlinearities that are not taken into account due to their intrinsic modeling complexity. An example would be the stall phenomenon that appears over a fraction of the wing area during the hovering phase.

The nonlinear model of the DarkO drone is represented by the following equations:

$$\begin{cases} \dot{p} = v \\ \dot{v} = \frac{1}{m} R(q) \sum F_b(x,u,w) + g \\ \dot{q} = \frac{1}{2} q \otimes \omega_b \\ \dot{\omega}_b = J^{-1} \sum M_b(x,u,w) - J^{-1} [\omega_b]_\times J \omega_b \end{cases} \tag{1}$$

where $x = \begin{pmatrix} p & v & q & \omega_b \end{pmatrix}^\mathsf{T}$ - state vector, $u = \begin{pmatrix} \omega_1 & \omega_2 & \delta_1 & \delta_2 \end{pmatrix}^\mathsf{T}$ - command vector, $p \in \mathbb{R}^3$ - position of the centre of mass in the inertial frame, $v \in \mathbb{R}^3$ - velocity, w - external wind perturbation, $g = \begin{pmatrix} 0 & 0 & g \end{pmatrix}^\mathsf{T}$ - (constant) gravity vector, q - unit quaternion characterizing the attitude, ω_b - rotational speed in the body frame, F_b and M_b - forces and moments matrices, m - mass and J - inertia matrix, R - rotation matrix, $p \otimes q$ - quaternion product, $[a]_\times b = a \times b$ - cross product conversion to matrix multiplication.

Quaternions are justified by the tilt-body nature of the UAV that calls for a global numerically stable formulation of attitude. The linearization is conducted around an equilibrium point that corresponds to a particular wind speed, obtaining a discrete set of LTI models. The equilibrium point for the trimming that corresponds to a wind speed equal to 0 is defined by the following system:

$$\begin{cases} \overline{p} = p_e \\ \overline{v} = 0 \\ \overline{q} = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \end{pmatrix}^T \\ \overline{\omega}_b = 0 \\ \overline{u} = \frac{mg}{(1 - \frac{S_{wet}}{4S_p} C_{d0})} \begin{pmatrix} 1 & 1 & 0 & 0 \end{pmatrix}^T \end{cases} \tag{2}$$

The linearized dynamic is characterized by the state vectors in (3) and by the state-space representation in (4). Note that $B = \begin{bmatrix} B_u & B_w \end{bmatrix}$ where B_u and B_w are the actuator and wind input matrices. For more details related to the nonlinear, linear dynamic modeling and equilibrium points of the DarkO drone, the reader is directed to the articles [5, 15].

$$\tilde{x} = \begin{pmatrix} \tilde{p} & v & \tilde{\epsilon} & \omega_b \end{pmatrix}^T = \begin{pmatrix} p - p_{eq} & v & \epsilon - \epsilon_{eq} & \omega_b \end{pmatrix}^T$$
$$\tilde{u} = u - u_{eq} \tag{3}$$

$$\dot{\tilde{x}} = A\tilde{x} + B\tilde{u}$$
$$y = C\tilde{x} \tag{4}$$

3 SOF CONTROLLER SYNTHESIS

The synthesis of a SOF controller poses a non-convex problem due to multiplications between decision variables, resulting in bilinear matrix inequalities. This complexity renders the optimization problem *NP-hard*. Referring to the BMI formulation in [16], various equivalent reformulations are conducted in [13], leading to the matrix inequality presented in Eq. 5. The proposed method here is extracted from [16]. For further insights into the methods employed, such as the S-variable approach and dual calculations, readers are directed to [16].

$$P \succ 0$$
$$He\left\{ \begin{bmatrix} 0 & 0 & P \\ 0 & 0 & 0 \\ P & 0 & 0 \end{bmatrix} \right\} \prec He\left\{ \begin{bmatrix} I \\ C \\ 0_{p-n,n} \\ -A \end{bmatrix} \left(\lambda \begin{bmatrix} I \\ C \\ 0_{p-n,n} \end{bmatrix} + M \right) \right] S_1 + \begin{bmatrix} 0 \\ S_2 \\ BZ \end{bmatrix} \begin{bmatrix} 0 & I & -H^T \end{bmatrix} \right\} \tag{5}$$

$$F = -ZS_2{}^{-1} \begin{bmatrix} I_p \\ 0_{n-p,p} \end{bmatrix} \tag{6}$$

Even though, this new reformulation is still characterized by non-convexity, it is of great interest because it has been proved that if a solution is found for the system in (5) when $\lambda = 1$, $M = 0$ and S_2 is non-singular then, a SOF gain matrix **F** can be computed, that guarantees closed-loop stability [13]. This gain matrix, not being directly optimized in the convex synthesis, is obtained using (6).

This proof serves as the overarching objective for the optimization process. Addressing the challenges stemming from the non-convex nature of problem 5 requires specific mathematical and technical advancements. A mathematical formulation introduced in the literature has shown promising initial theoretical results. The algorithm, to be detailed in the following subsection, will be applied to the DarkO drone's dynamics. The resulting analysis will draw conclusions regarding the algorithm's effectiveness in practical test scenarios.

3.1 Deterministic Iterative Algorithm for SOF Design

The problem of synthesising a SOF gain, is characterized by its intrinsic non-convexity nature that is accompanied by computation difficulties related to *NP-hard* problems. This, coupled with an inherent complexity that comes from the impossibility of optimizing all of the decision variables simultaneously, makes the process of splitting the non-convex problem into multiple auxiliary convex subproblems extremely appealing. The bilinear matrix inequality (5) will be employed to translate these goals into an optimization framework. Each particular goal is attained by optimally fixing a part of the decision variables, while refining the rest through an iterative process. For designing a SOF controller, a 3-phase algorithm introduced in [13] will be implemented Fig. 2 that takes as input the **A**, **B**, **C** matrices of the plant's linearized dynamics.



Figure 2: Structure of Optimization Algorithm

3.1.1 Initialization Phase

The objective of the first phase is essentially to provide an initial estimate for a state feedback gain matrix controller **K** that stabilizes the closed loop $\dot{x} = (A + BK)x$. For this, the

http://www.imavs.org/

strategy taken is to fix the decision variables λ, M, H present in (5) in the following way:

$$\begin{aligned} \lambda &= \lambda = 0 \\ M &= M_0 = \begin{pmatrix} C^{\circ}C \\ C^{\perp T} \end{pmatrix} \\ H &= H_0 = JM_0^{-1} \\ J &= (-\mu - h)I, \end{aligned} \tag{7}$$

where, h is a positive scalar and $-\mu$ is the maximum real part of the eigenvalues of A. The feasibility study that is conducted in the initialization phase is the following:

$$P \succ 0$$
$$He\left\{\begin{bmatrix} 0 & 0 & P \\ 0 & 0 & 0 \\ P & 0 & 0 \end{bmatrix}\right\} \prec He\left\{\begin{bmatrix} I \\ -M_0 \\ -A \end{bmatrix} S_1 + \begin{bmatrix} 0 \\ S_2 \\ BZ \end{bmatrix} \begin{bmatrix} 0 & I & -H_0^T \end{bmatrix}\right\} \tag{8}$$

If (8) is feasible for a combination of the decision variables S_1, S_2, Z and for the Lyapunov certificate P, then an initial guess of a stabilizing state feedback gain **K** is found, and one can pass to the iteration phase. On the other hand, if a solution is not found, the parameter **h** is increased by 1 and the initialization phase is executed again. This phase solves the (SI) and (SF) problems. The outputs of this phase are:

$$\begin{aligned} S_{1.0} &= S_1 \\ \hat{K}_0 &= -ZS_2^{-1} \\ K &= -ZS_2^{-1}M_0 \end{aligned} \tag{9}$$

3.1.2 Iteration Phase - Step k,1

This phase comprises two iterative steps. In the first step, the optimization variables $\boldsymbol{\lambda}$ and **M** in 5 previously fixed during initialization, are now treated as decision variables. To maintain convexity, the slack variable S_1 is set at the value $\boldsymbol{S_{1,0}}$ determined in the initialization phase. Then, system (10) is solved.

$$P \succ 0, \quad \begin{bmatrix} (1-\lambda)I & M^T \\ M & I \end{bmatrix} \geq 0, \quad \lambda \geq 0$$
$$He\left\{\begin{bmatrix} 0 & 0 & P \\ 0 & 0 & 0 \\ P & 0 & 0 \end{bmatrix}\right\} \prec He\left\{\begin{bmatrix} I \\ -\left(\lambda \begin{bmatrix} C \\ 0_{p-n,n} \end{bmatrix} + M\right) \\ -A \end{bmatrix} S_{1,k-1} + \begin{bmatrix} 0 \\ -I \\ B\hat{K}_{k-1} \end{bmatrix} \begin{bmatrix} 0 & -S_2 & Y^T \end{bmatrix}\right\} \tag{10}$$

In (10), two matrix inequalities appear, in addition to the Lyapunov certificate inequality and the inequality (5). These additional constraints, contribute towards achieving the objective of the iteration phase which is to find a solution for (5) where the variables $\boldsymbol{\lambda}$ and **M** converge to the values of 1 and 0, respectively, this being a mandatory requirement for meeting the main optimization goal. This is accomplished by setting the maximization of $\boldsymbol{\lambda}$ as an objective in the optimization solver. The second matrix inequality works on constraining the norm of **M** and decreasing it to 0, as $\boldsymbol{\lambda}$ increases to 1. The third inequality is merely a constraint on $\boldsymbol{\lambda}$ to be positive.

If a solution is found for (10) and if for this solution $1-\lambda$ is lower than a specific tolerance, this means that one can pass on to the third and last phase, namely the validation. The outputs for this phase are:

$$\begin{aligned} \lambda_k &= \lambda \\ M_k &= M \\ H_k^T &= {S_2}^{-1}Y^T \end{aligned} \tag{11}$$

If not, there is a second step in the iteration phase.

3.1.3 Iteration Phase - Step k,2

Step **k,2** is essentially an initialization step in disguise. Similar to the first phase, where λ, M and H were fixed as inputs to generate a stabilizing feedback gain **K**, the second step fixes the same decision variables with updated values from step **k,1**.

The optimization problem that needs to be solved in this phase is the following:

$$P \succ 0$$
$$He\left\{\begin{bmatrix} 0 & 0 & P \\ 0 & 0 & 0 \\ P & 0 & 0 \end{bmatrix}\right\} \prec He\left\{\begin{bmatrix} I \\ -\hat{M}(\alpha) \\ -A \end{bmatrix} S_1 + \begin{bmatrix} 0 \\ S_2 \\ BZ \end{bmatrix} \begin{bmatrix} 0 & I & -H_k^T \end{bmatrix}\right\} \tag{12}$$
$$\hat{M}(\alpha) = \left((1+\alpha(\lambda_k - 1)) \begin{bmatrix} C \\ 0_{p-n,n} \end{bmatrix} + \alpha M_k\right) \tag{13}$$

In this step, a new term, $\hat{M}(\alpha)$ appears, dependent on a new decision variable α. The goal is to minimize α using the bisection method, converting the non-convex problem into a convex one. Ideally α converges to 0, signaling the end of this iteration phase and progression to the final validation phase. If $\alpha \approx 0$ within a certain tolerance, (13) simplifies to (14) and as a result, the values of the decision variables for the solution that has been found for (12) that correspond to the step **k,2** are also values for which (5) is solved when particularized for $\lambda = 1$ and M = 0. Finding a solution for (5) with $\lambda = 1$ and M = 0, enables the computation of a stabilizing SOF controller.

$$\hat{M}(\alpha) = \left((1+\alpha(\lambda_k - 1)) \begin{bmatrix} C \\ 0_{p-n,n} \end{bmatrix} + \alpha M_k\right) \overset{\alpha \approx 0}{=} \begin{bmatrix} C \\ 0_{p-n,n} \end{bmatrix} \tag{14}$$

If at step **k,2** α is not smaller than a fixed tolerance, the algorithm will start a new iteration step at **k,1**. The outputs of the current phase are:

$$\begin{aligned} \alpha_k &= \alpha \\ \hat{K}_{k-1} &= -ZS_2^{-1} \\ S_{1,k} &= S_1 \end{aligned} \tag{15}$$

3.1.4 Validation Phase

In cases where the iteration stage provides solutions for which $\boldsymbol{\lambda}$ and **M** are exactly equal to 1 and 0, a validation phase is not required because a SOF controller can be computed

http://www.imavs.org/

directly. In practice, the algorithm exits the iteration phase with solutions for $\boldsymbol{\lambda}$ and $\mathbf{M}$ close to 1 and 0, due to limitations of the optimization solvers and numerical accuracies. In these cases, the conditions for achieving the main optimization objective are not exactly met. As a result, a validation phase is necessary, where the problem to be solved is defined by (16) where the variables $\boldsymbol{\lambda}$ and $\mathbf{M}$ are set to 1 and 0 and $\mathbf{H}$ is set to H_k (obtained during the iteration phase). If a solution is found, the gain matrix $\mathbf{F}$ can be constructed with (6) that is guaranteed to stabilize the closed loop (A+B**F**C). This last phase, provides the solution to the (OI) and (OF) goals.

$$P \succ 0$$
$$He\left\{\begin{bmatrix} 0 & 0 & P \\ 0 & 0 & 0 \\ P & 0 & 0 \end{bmatrix}\right\} \prec He\left\{\begin{bmatrix} I \\ -\begin{bmatrix} C \\ 0_{p-n,n} \end{bmatrix} \\ -A \end{bmatrix} S_1 + \begin{bmatrix} 0 \\ S_2 \\ BZ \end{bmatrix} \begin{bmatrix} 0 & I & -H^T \end{bmatrix}\right\} \tag{16}$$

3.2 *Plant Augmentation and Control Architecture*

The control architecture for the DarkO drone is designed to stabilize it during hovering, both with and without external disturbances. For example, to counteract a headwind affecting linear velocity along the $x_{[b]}$ and $z_{[b]}$ axes and generating a moment about the $y_{[b]}$ axis, the control scheme uses the ailerons and propellers symmetrically to generate a compensatory moment and force. Integral action is employed to counteract the force along the $z_{[b]}$ axis, ensuring asymptotic convergence to the desired force using two integrators, one for the motors and one for the elevons. Other common control methods for this drone configuration include PID scheduling, decoupled speed and attitude controllers, and nonlinear controllers with command allocation. However, none of these approaches directly address wind effects.

The synthesis of a SOF controller for the augmented DarkO dynamics takes its inspiration from [10] where it is shown that a dynamic output compensator of order $q \leq n$ (n-plant order), can be converted to a static output feedback through state-space augmentation. In our case, dynamic terms such as integrators and filters are added to the plant dynamics, resulting in an augmented state-space for which a static feedback law comprised of a pre-compensator static gain matrix will be synthesised. The implemented architecture, presented in Fig. 3, is a reformulation of the control law presented in [17]. It is composed of the linearized DarkO drone dynamic block, the saturation block for the two propellers and the two elevons that introduces limitations for the angular speed and angular rate of the electrical motors as well as angular deflection and angular rate for the control surfaces. The introduction of the actuator dynamics is subject for future studies. The output selection matrix eliminates the measurement of the pitch angle state, $\boldsymbol{\theta}$.

For control design, a PI-like controller structure is employed to minimize static error and enhance rejection of external disturbances. This is done by adding pre-compensator filters at the input of the system, based on

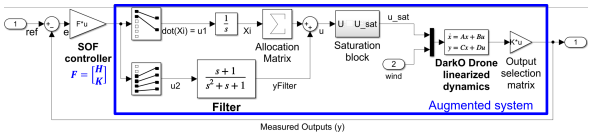


Figure 3: Plant Augmentation

the open-loop shaping methodology [14] where an integrator is added to ensure performance (high gain) at low frequency, which corresponds to the tracking specification, and a second-order filter is added to ensure that the controller is strictly proper and to be robust with respect to neglected dynamics thanks to a roll-off behaviour. The filter's cutoff frequency is set at approximately 5 Hz to align with closed-loop dynamics and reject white noise in measured signals. For limiting the number of integrators to 2 (one that generates the integrative command for the two propellers and the other one for the two elevons) the outputs of the integrator block are doubled by an allocation matrix $\sum$. Implementing a second order filter comes with the extra benefit of avoiding a direct feed-forward term, that may amplify unwanted sensor noise. The SOF matrix $\mathbf{F}$ comprises the gain matrices $\mathbf{H}$ and $\mathbf{K}$ for the integrative and proportional action, respectively. The equations and the state-space representation of the augmented plant are found in (17) and (18).

$$\begin{gathered} \dot{x}_i = u_1 \\ u = \sum x_i + y_{filter} = \sum x_i + Filter \cdot u_2 \\ \dot{x} = Ax + Bu = Ax + B(\sum x_i + y_{filter}) = Ax + B\sum x_i + BCx_{filter} \\ \sum = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = Fe = F\left(\begin{bmatrix} ref \\ \not{r}_{8\times 1} \end{bmatrix} - y\right) \end{gathered} \tag{17}$$

$$\begin{gathered} \begin{pmatrix} \dot{x} \\ \dot{x_c} \\ \dot{x}_{filter} \end{pmatrix} = \begin{bmatrix} A & B\sum & BC_{filter} \\ 0 & 0 & 0 \\ 0 & 0 & A_{filter} \end{bmatrix} \begin{pmatrix} x \\ x_c \\ x_{filter} \end{pmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & B_{filter} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \\ y = \begin{bmatrix} C & 0 & 0 \end{bmatrix} \begin{pmatrix} x \\ x_c \\ x_{filter} \end{pmatrix} \end{gathered} \tag{18}$$

Where $x_i \in \mathbb{R}^2$ - integrative states, $x_f \in \mathbb{R}^4$ - filter states, $r \in \mathbb{R}^3$ - reference signal, $y \in \mathbb{R}^{11}$ - measured states and $u \in \mathbb{R}^4$ - plant input.

4 RESULTS

4.1 *Simulation Results*

The approach from section 3.1 was applied to the augmented system from section 3.2, focusing on the linearized dynamics for a wind speed of 0 (hovering scenario). This resulted in four stabilizing controllers for h=12,13,14, and 15, with h varying between 1 and 40. Multiple iterations with different h values, serving as various initialization points for the optimization algorithm, are necessary to enhance the likelihood of convergence.

Figure 4 shows the closed-loop temporal response of the four controllers, with reference steps for the x-axis

http://www.imavs.org/

at 5s, y-axis at 90s, and z-axis at 40s. All controllers effectively stabilize the closed-loop dynamics and track position references on the XYZ axes. The response is slow on the X and Z axes and fast on the Y axis. Notably, a position step on one axis does not cause significant offset or change on the other two axes, indicating strong decoupling between the XYZ position states. Additionally, the actuators were never saturated.

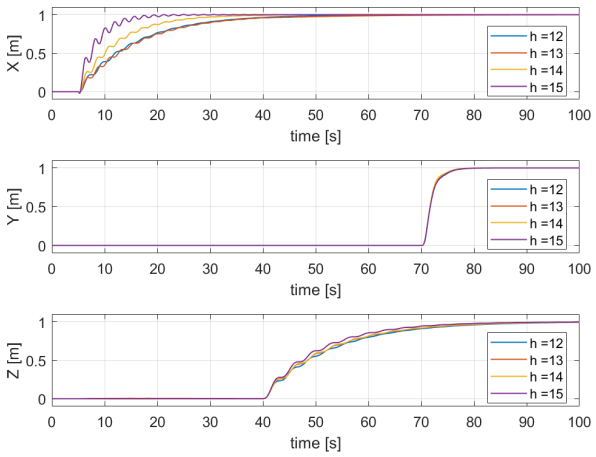


Figure 4: Closed-Loop response of SOF controllers - Linear Dynamics

The first subplot of Fig. 5 shows that the state $\boldsymbol{\theta}$ experiences increasing oscillations as h increases. While the pitch angle $\boldsymbol{\theta}$ is not directly controlled, it naturally converges to 0 as the other states stabilize, indicating hovering flight without wind. The second subplot demonstrates that for dynamics linearized around a specific wind speed, $\boldsymbol{\theta}$ converges to a non-zero value, compensating for external disturbances.

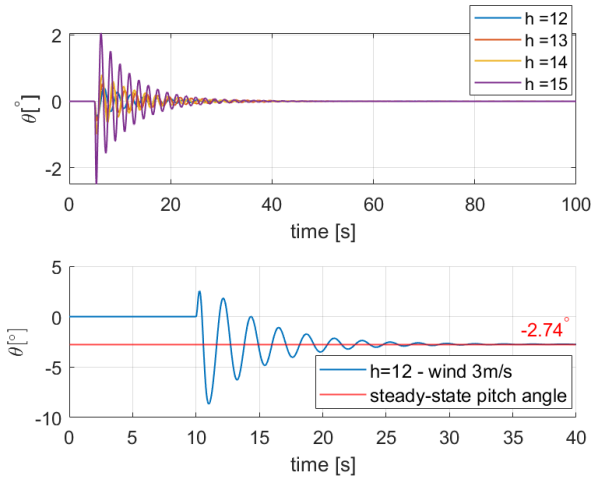


Figure 5: Closed-Loop temporal response - theta state

For analysis purposes, Eq. 19 and 20 present the gains of the synthesized controller **F** (block illustrated in 3) comprising the matrix gains **H** and **K** for h=12 and linearized dynamics at 0 wind speed. By taking a more attentive look at the gain matrix **K** that outputs 4 command signals used for the asymmetric or symmetric control of the drone actuators, one can observe that for the 1st and 2nd rows (that generate a demand of thrust of the two propellers) as well as for 3rd and 4th rows (that generate a demand for a deflection angle for the two elevons) are either nearly equal or opposite, with few exceptions. This pattern emerged naturally from the optimization process without any constraints, reflecting an intuitive and coherent alignment with the drone's dynamics.

$$H = \begin{bmatrix} 3.5e-06 & -3.0e-07 & 1.5e-02 & -6.9e-06 & -7.4e-06 & 4.4e-01 \\ -2.0e-02 & -7.5e-06 & 1.6e-05 & -2.0e-01 & 4.4e-05 & -1.1e-04 \\ 1.8e-04 & 1.1e-04 & -8.4e-05 & -8.6e-05 & -2.3e-04 \\ -9.2e-05 & -9.1e-05 & 7.4e-06 & 7.6e-01 & -3.9e-05 \end{bmatrix} \quad (19)$$

$$K = \begin{bmatrix} 7.6e-03 & -2.0e+01 & 9.9e+00 & 3.1e-02 & -4.5e+01 & 8.2e+01 \\ 5.4e-03 & 2.0e+01 & 9.9e+00 & 8.8e-03 & 4.5e+01 & 8.2e+01 \\ -4.2e+00 & -3.2e-01 & 8.7e-03 & -4.5e+01 & -4.4e-01 & -3.1e-02 \\ -4.2e+00 & 3.2e-01 & 4.8e-03 & -4.5e+01 & 4.3e-01 & 3.3e-02 \\ -3.1e+02 & -3.1e+02 & 6.4e-01 & 3.0e-03 & -1.0e+02 \\ 3.1e+02 & 3.1e+02 & -6.5e-01 & -1.9e-03 & 1.0e+02 \\ -1.2e+01 & 1.6e+00 & -3.1e+01 & 3.9e+01 & -1.6e+00 \\ 1.2e+01 & -1.7e+00 & 3.1e+01 & 3.9e+01 & 1.6e+00 \end{bmatrix} \quad (20)$$

4.2 *Experimental Results*

For experimental validation, the designed controllers and architecture were tested on the real DarkO drone model built at ENAC Fig. 6. DarkO is assembled from multiple 3D printed Onyx parts (a robust material comprising omnidirectional carbon fibres). Flight tests took place in ENAC's volière, equipped with an Optitrack motion capture system providing position and attitude data at 40Hz, eliminating the need for a GPS sensor. Speed is obtained by a finite difference between the position. Data fusion algorithms, including Invariant Filters [18, 19], combined data from Optitrack (position, speed and attitude) and the DarkO drone's IMU unit for improved state estimation. This estimate is used to create the output vector y used by the control law (17). Given the architecture of the control loop, the estimation must be of very high quality with the lowest possible delay. Paparazzi UAV open-source software and hardware packages were utilized [20] and the modular nature of Paparazi permit to use the existing implementation of the invariant filter to estimate the DarkO state vector and to add a stabilisation module based on the control law described above (see 3.2). This software is embedded in an autopilot designed and manufactured at Enac, an Apogee autopilot board[1] for processing. The autopilot board sampled command and control laws at 500Hz, generating appropriate control commands to achieve the desired flight manoeuvres and store all the data for posterior analysis.

The following experimental flight tests mark the first and only successful simulations in ENAC's volière where controllers relying exclusively on model-based synthesis methods were used to control a convertible drone. Prior

[1]Product data sheet - `https://wiki.paparazziuav.org/wiki/Apogee/v1.00`

http://www.imavs.org/

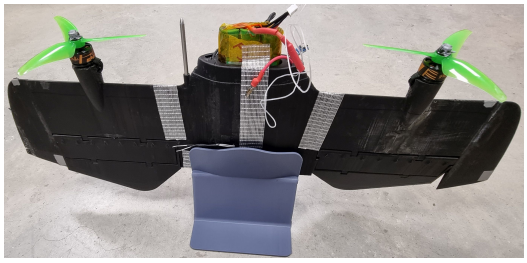


Figure 6: DarkO Drone experimental model

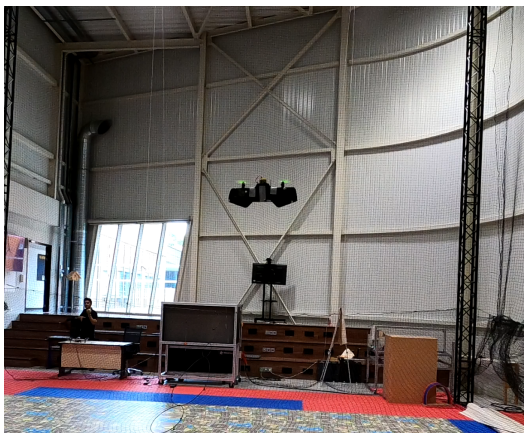


Figure 7: DarkO drone during an experimental flight test

flights for this drone type utilized combined model– and sensor–based control designs like INDI algorithms. Each flight starts with the drone taking off and stabilizing at a reference position using an INDI controller. After reaching the desired hovering location, the INDI controller is replaced with the model-based controller. Figure 8 shows the flight test results of one of the four synthesized controllers for the designed control architecture. Figure 7 depicts the DarkO drone during its test flight. All four controllers were tested and validated on the real system, demonstrating stable performance and accurate position tracking on all three axes.

As it was also observed in the simulations, on the X and Z axes, the time response is relatively high whereas on the Y axis, the time response is drastically quicker, due to the strong differential actuation on this axis backed up by the fast actuator dynamics. On the Z axis, the switch from the INDI controller to the synthesised controller, results in a relatively important vertical descent of the drone. As a consequence, further tuning and modeling needs to be performed in order to correct the state and command input initialization values corresponding to the equilibrium point. The dynamic of the drone on Z is characterized by small oscillations around the reference position, that can also be observed in the command signal of the angular speeds of the propellers, illustrated in Fig. 8. These reduced oscillations are due to the actuator dynamics not being considered in the control law. This problematic will be studied in the

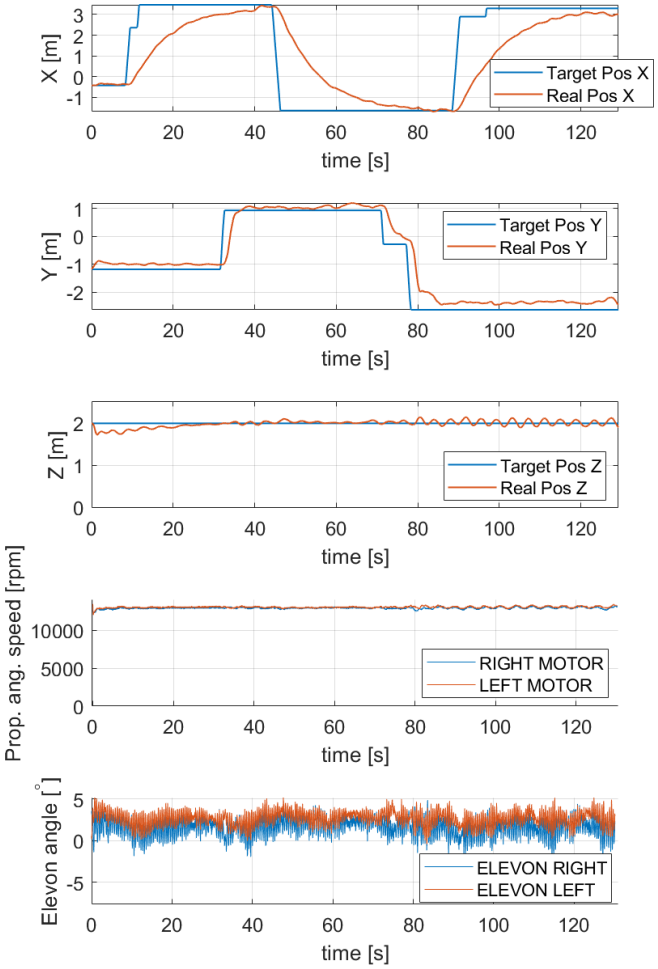


Figure 8: Flight Test, Closed-Loop Response and Command Input, h = 12

future, particularly with the use of a motor speed controller (see AM32-MultiRotor-ESC-firmware [2]). It's worth noting that the optimization algorithm's primary objective is to achieve closed-loop stabilization, disregarding performance or robustness criteria. During experimental flights, the controller did not saturate the actuators (see Fig. 8).

5 CONCLUSIONS

A convex optimization algorithm, utilizing the LMI framework and Lyapunov's stability theory, was employed to synthesize SOF controllers for the DarkO convertible drone model. This model-based synthesis technique effectively stabilized the closed-loop dynamics of the augmented plant, ensuring satisfactory temporal response and reference tracking without actuator saturation. Despite incomplete modeling of nonlinear phenomena, the controllers demonstrated robustness during initial experimental

[2] https://github.com/FlorianSan/AM32-MultiRotor-ESC-firmware

http://www.imavs.org/

demonstrations on the DarkO drone model. With the designed SOF controller matrices and command law structure, successful experimental flights were conducted for hovering and trajectory tracking. This outcome serves as a solid proof of concept for the developed control law in terms of performance and robustness.

REFERENCES

[1] Guillaume J.J. Ducard and Mike Allenspach. Review of designs and flight control techniques of hybrid and convertible vtol uavs. *Aerospace Science and Technology*, 118:107035, 2021.

[2] Jacson M. O. Barth, Jean-Philippe Condomines, Murat Bronz, Leandro R. Lustosa, Jean-Marc Moschetta, Cédric Join, and Michel Fliess. Fixed-wing uav with transitioning flight capabilities : Model-based or model-free control approach? a preliminary study. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1157–1164, 2018.

[3] Yijie Ke, Kangli Wang, and Ben M. Chen. Design and implementation of a hybrid uav with model-based flight capabilities. *IEEE/ASME Transactions on Mechatronics*, 23(3):1114–1125, 2018.

[4] Yijie Ke, Kangli Wang, Kehong Gong, Shupeng Lai, and Ben M. Chen. Model based robust forward transition control for tail-sitter hybrid unmanned aerial vehicles. In *2017 13th IEEE International Conference on Control and Automation*, pages 828–833, 2017.

[5] Florian Sansou and Luca Zaccarian. On local-global hysteresis-based hovering stabilization of the darko convertible uav. In *2022 European Control Conference (ECC)*, pages 40–45, 2022.

[6] Jacson M. O. Barth, Jean-Philippe Condomines, Jean-Marc Moschetta, Aurélien Cabarbaye, Cédric Join, and Michel Fliess. Full model-free control architecture for hybrid uavs. In *2019 American Control Conference (ACC)*, pages 71–78, 2019.

[7] Rasmus Steffensen, Agnes Steinert, and Ewoud J. J. Smeur. Nonlinear dynamic inversion with actuator dynamics: An incremental control perspective. *Journal of Guidance, Control, and Dynamics*, 46(4):709–717, apr 2023.

[8] O. Toker and H. Ozbay. On the np-hardness of solving bilinear matrix inequalities and simultaneous stabilization with static output feedback. In *Proceedings of 1995 American Control Conference - ACC'95*, volume 4, pages 2525–2526 vol.4, 1995.

[9] J.C. Doyle, K. Glover, P.P. Khargonekar, and B.A. Francis. State-space solutions to standard h/sub 2/ and h/sub infinity / control problems. *IEEE Transactions on Automatic Control*, 34(8):831–847, 1989.

[10] V.L. Syrmos, C.T. Abdallah, P. Dorato, and K. Grigoriadis. Static output feedback—a survey. *Automatica*, 33(2):125–137, 1997.

[11] B. Anderson, N. Bose, and E. Jury. Output feedback stabilization and related problems-solution via decision methods. *IEEE Transactions on Automatic Control*, 20(1):53–66, 1975.

[12] P. Apkarian and D. Noll. Nonsmooth h-infinity synthesis. *IEEE Transactions on Automatic Control*, 51(1):71–86, 2006.

[13] D. Arzelier, F. Dabbene, S. Formentin, D. Peaucelle, and L. Zaccarian. *Robust Static Output Feedback Design with Deterministic and Probabilistic Certificates*, pages 121–148. Springer International Publishing, Cham, 2018.

[14] D. McFarlane and K. Glover. A loop-shaping design procedure using h infinity synthesis. *IEEE Transactions on Automatic Control*, 37(6):759–769, 1992.

[15] Florian Sansou. Commande hybride d' un drone convertible pour des déplacements sous optimaux. *ArXiv*, abs/2203.15387, 2022.

[16] Yoshio Ebihara, Dimitri Peaucelle, and Denis Arzelier. *S-Variable Approach to LMI-Based Robust Control*. Springer, January 2015.

[17] Florian Sansou, Fabrice Demourant, Gautier Hattenberger, Thomas Loquen, and Luca Zaccarian. Open wind tunnel experiments of the darko tail-sitter longitudinal stabilization with constant wind. *IFAC-PapersOnLine*, 55(22):1–6, 2022. 22nd IFAC Symposium on Automatic Control in Aerospace ACA 2022.

[18] Jean-Philippe Condomines, Cédric Seren, and Gautier Hattenberger. Nonlinear state estimation using an invariant unscented Kalman filter. In *AIAA GNC 2013, AIAA Guidance, Navigation and Control Conference*, pages pp 1–15 ; ISBN : 978–1–62410–224–0, Boston, United States, August 2013. AIAA.

[19] Jean-Philippe Condomines, Cédric Seren, and Gautier Hattenberger. Pi-invariant unscented kalman filter for sensor fusion. In *53rd IEEE Conference on Decision and Control*, pages 1035–1040, 2014.

[20] Gautier Hattenberger, Murat Bronz, and Michel Gorraz. Using the Paparazzi UAV System for Scientific Research. In *IMAV 2014, International Micro Air Vehicle Conference and Competition 2014*, pages pp 247–252, Delft, Netherlands, August 2014.

http://www.imavs.org/

A Study of Gain Tuning for Prolonged Physical Contact of Fully-Actuated UAVs

Chantelle Singh, Shahab Kazemi, Jonty Kirk, Katrina Chan, and Karl Stol
Department of Mechanical and Mechatronics, University of Auckland, New Zealand.
Corresponding author Email: shahab.kazemi@auckland.ac.nz

http://www.imavs.org/

ABSTRACT

This study explores achieving stable, prolonged physical contact with a fully-actuated multi-rotor unmanned aerial vehicle (UAV). Through rigorous experiments, we analyze the dynamic interplay between an octocopter UAV and its environment, particularly concerning the influence of control gains on system behaviour. Employing the widely-used P-PID controller, the paper unravels the implications of vibrations and instabilities during UAV operation. Using five different angles, the research elucidates how tuning P-PID gains based on contact angle and dynamic coupling impacts the UAV's response to external forces, enhancing stability and overall performance. Moreover, the effect of the center of mass (CoM) location is also considered a factor affecting the experiments. The results show that the response of the UAV after tuning is capable of robust contact testing.

1 INTRODUCTION

Unmanned aerial vehicles (UAVs) applications include but are not limited to surveying, mapping, inspecting, plant and wildlife preservation, agriculture, and atmospheric science monitoring, as detailed in various studies [1, 2, 3, 4, 5]. In this context, emerging technologies have made it inevitable for UAVs to demonstrate robust performance and stability in executing physical interactions, such as grasping, sampling, contact-based inspecting, transporting, and assembling. A UAV application involves two-way physical interaction, where the UAV applies force and receives reactions from its surroundings. These interactions can significantly affect the UAV's behavior.

UAVs conducting physical interaction is called aerial manipulation, which is manipulating the environment in mid-air. Aerial manipulation is of recent interest due to the several applications it is suited to. Some examples of aerial manipulation applications are shown in Figure 1.

There are many interesting studies related to UAVs' physical interactions. A study introduced and validated an armed-UAV for valve manipulation. Using a human-machine interface, a single operator directed the UAV to approach, grasp, and rotate an industrial valve using arm motion and voice control. The system demonstrated exceptional stability and resilience to contact forces [6]. In another study a 6-m-tall tower of 1500 foam modules was constructed in 18 hours using four autonomous quadcopters [7]. For precise aerial repair, explicitly focusing on sealing and filling cracks, an integrated 3-degrees of freedom (-DoF) manipulator capable of compensating for both translational and rotational offsets attached to the UAV frame was proposed. The outcomes highlighted potential applications for accurate aerial inspections and repairs in demanding environments, such as nuclear or petrochemical plants [8]. The UAVs interactions with trees for canopy sampling were discussed in [9]. The sampling procedure was segmented into four phases: unimpeded flight to reach the branch, actively tracking the branch's position, coupling and decoupling from the branch, and finally, departing from the canopy. An octocopter configuration incorporated a novel manipulator, enabling robust contact operations for precise inspections [10]. The manipulator, rotating around the UAV's CoM, oriented the end effector within 180 degrees of the workspace, efficiently rejected external perturbations, maintained a specific contact force against the surface, and calculated the end effector's position concerning the aerial robot.

Another application is the use of UAVs for inspecting infrastructure, specifically power transmission assets, which has garnered significant attention due to its potential to improve safety and cost-effectiveness in the energy transmission sector [1]. Powerline contact testing is preferable to non-contact testing as quantitative information such as voltage drop, current, and resistance can be found by physically testing the powerline using a probe[1] (visual power line testing can use thermal imaging to detect overheating conductors and regular imaging to detect potential structural damage to cables). Ikeda et al. researched the flight control of an under-actuated UAV contact testing a bridge with a single DoF manipulator [11]. The manipulator used was a rigid connection extended horizontally from the UAV. The UAV tested the bridge for structural faults by sending ultrasonic pulses through the structure.

Similar to the powerline contact testing, there are certain forces that the controller has to deal with. Through experimental analysis, it was observed that the pitch angle of the UAV during contact with the bridge wall is closely correlated

[1] https://www.scribd.com/document/428682812/SensorLink-Catalogue-pdf

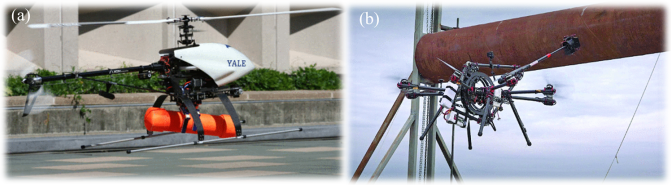


Figure 1: Aerial manipulation examples. (a) Helicopter carrying an object [13]. (b) UAV with manipulator for inspection at industrial sites [10]

with the corresponding contact force experienced by the UAV. They used classical control in the form of cascaded proportional, integral, and derivative (PID) control to keep the UAV stable while it was in contact with the bridge. When UAVs deliberately interact with the environment, the standard position control architecture for a free-flying UAV is no longer robust. Tomic and Haddadin [12] considered this control problem in creating a unified framework for UAV interaction control. They proposed using admittance and impedance controllers in developing control for UAVs to allow for compliance with obstacles allowing for less rigid contact.

This study aims to achieve stable contact testing with a fully-actuated UAV and a rigidly attached arm by systematically tuning the free-flight gains based on different angles through which it made the physical contact. The fully-actuated UAV can perform pure translation without having to tilt (translating and maintaining flight stability without having to pitch or roll) while providing a horizontal thrust. The ability to exert force without titling makes the fully-actuated UAV suitable for rigid arm attachments. The rest of this article is organized as follows.

In Section 2, we detail the physical setup, including the fully-actuated UAV configuration and specific contact attachments, establishing a foundation for the experimental context. Additionally, Section 2 dives into software and hardware integrations, ensuring the readiness of the setup. Section 3 shifts focus to the control structure, explaining the difference between the used controller and the standard PX4 algorithm. Section 4 presents gains tuning tailored for experimental scenarios, results, detailing the UAV's performance during prolonged physical interactions and responses to varied contact angles. The accompanying analysis provides deeper insights. Section 5 concludes with a concise summary of contributions and outlines potential future research directions.

2 UAV AND EXPERIMENTAL SETUP DESCRIPTION

The fully-actuated UAV used in this paper is an octocopter consisting of four rotors on the top and four rotors on the bottom (see Figure 2). Each of these rotors is fixed at 31º, allowing for translation in the X and Y directions without pitching or rolling (flying level). This characteristic is essential because the physical interaction mechanism consists of an arm and end effector (inspired by the prob from Sen-

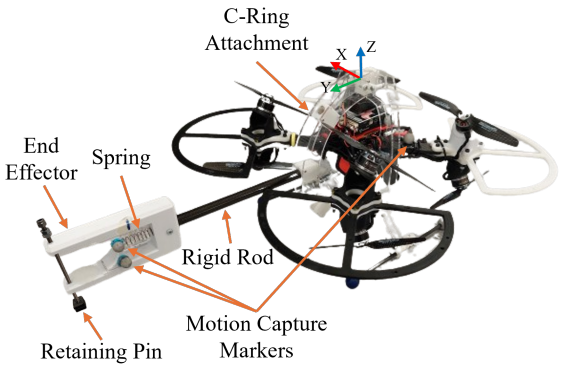


Figure 2: Fully-actuated octocopter used. X, Y, Z are body frame coordinates

sorLink[2]) rigidly attached to the UAV at specific angles, requiring the UAV to fly level when making contact with its surroundings. As the external environment, a long horizontal clamped-clamped rod with a stiffness of 508 N/m a is used (0.5 m above the ground). During each experiment, the UAV must exert a force on the rod while remaining level to ensure that it applies the force at a specific angle (0º, 45º, 90º, -45º, or -90º). If the UAV changes its attitude such that it no longer has minimal roll and pitch, it is challenging to get stable contact with sufficient force being applied by the UAV as it will slip or flip around the rod. In order to create a rigid attachment between the arm and the UAV, a C-shaped rig is designed (laser cut out of a 6 mm acrylic sheet), allowing for securing a carbon fiber arm (low weight and high strength) to the airframe by five different angles in a vertical body-fixed plane shown in Figure 3. The C-shaped rig is placed in a way that has no interference with the propellers, allowing for a safe flight. The rig is firmly attached at all points with additional mounting brackets to ensure rigid connections and less vibration.

In this paper the end effector, including the attached carbon fibre rod, has a total mass of 43 g. A safety retaining pin, which has minimal effect on the experimental results, is added as a precaution in case something goes wrong. In addition, a spring is embedded into the end effector to determine the contact force by measuring its compression. Figure 4 shows the calibration result using an analogue force gauge to find the relationship between the force and displacement of the spring. The spring's displacement is readily measurable by adding two reflective markers and using the motion capture system. The approach provides a straightforward method to calculate the force exerted from the end effector to the clamped rod instead of force sensors, which may add significant complexity due to increased weight, communica-

[2]https://www.scribd.com/document/428682812/SensorLink-Catalogue-pdf

http://www.imavs.org/

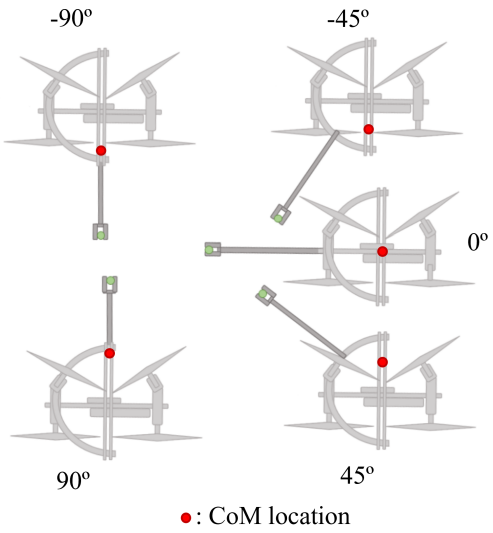


Figure 3: Contact angles showing CoM location

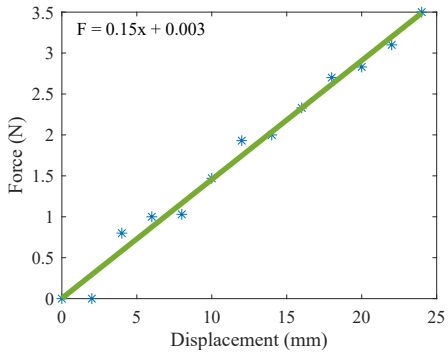


Figure 4: Calibration graph for the end effector's spring, based on data captured from bench handheld experiments

tion challenges, and integration with the flight controller. Due to all the mentioned physical modifications, the mass of the UAV is increased from 806 g to 977 g.

Since this research only focuses on physical interaction, as opposed to tracking a trajectory and making physical contact, a safety retaining pin and tether is be used during the experiment. The tether was attached around the top of the UAV and set up to be extremely loose. Note that the impacts of these safety features are minimal and only in effect during arming and disarming portions of the flight and not during contact.

All the experiments are conducted in a motion capture laboratory consisting of several cameras to track the movement of reflective markers attached to specific parts of the UAV. Also, a Simulink platform is used to send the setpoints required by the offboard controller to the UAV. The offboard controller is a position controller that can send position and attitude setpoints to the UAV. The controller is uploaded on the Pixhawk 4 Mini flight controller (attached to the UAV), which uses an ESP32 microcontroller to communicate with an offboard computer using the MAVLink protocol. Figure 5 shows the relationships between the hardware elements of the experiment.

After turning on the motion capture system, the PC used for generating the set points is connected to the UAV Wi-Fi module and then runs the Simulink interface. In the next step, the motion capture readings must correlate with the sensor readings from the Pixhawk 4 on the UAV through the EKF2 module. After ensuring that the motion capture data matches the onboard Pixhawk 4 sensors, the UAV manually takes off using the transmitter. Immediately after the take-off, the pilot switches to the offboard controller to start regulating toward the desired setpoints. The setpoint regulation is executed through the Simulink interface, sending high-level setpoints for position. Note that the desired set points, which are the position of the UAV relative to the contact point, are determined previously using motion capture markers by placing the unarmed UAV in the required location for contact and analysing the vicon tracker readings. After finishing the prolonged contact, the human pilot disarms the UAV. The last step involves collecting the flight log and motion capture recording data, which is then used to calculate the displacement of the spring on the end effector to calculate the coaxial compression force. This procedure is shown as a flowchart in Figure 6.

3 CONTROL STRUCTURE

The utilized controller is a customized iteration of the standard P-PID controller integrated with the PX4 Autopilot version 1.12.3[3]. The standard P-PID involves a PID attitude rate controller, the innermost control loop, and a P controller for attitude (Euler angles). Additionally, we have the velocity PID controller and the position P controller. In this paper, we focus on the position and velocity of P-PID loops. In standard PX4 Autopilot, the translational commands from position control are converted to angle setpoints and passed to the attitude controller (in a cascaded form). In this work, however, the attitude controller runs parallel to the position controller, with constant zero setpoints for angles to keep the UAV level during the flight. Figure 7 shows the block diagram of the used controller.

As mentioned before, the focus of this work is exploring the P-PID gains and their effect on the physical interaction mission. A systematic approach to modifying the controller gains is taken by initially tuning the control gains for free flight. To begin this process, the attitude rate controller is tuned first since it is more sensitive. Tuning involves flight tests after incremental gain changes and response analyses. Once the rate controller is tuned with acceptable root mean square error (RMSE) values, the attitude, velocity, and position controllers are tuned. These free flight gains provide the

[3]https://docs.px4.io/v1.12/en/

http://www.imavs.org/

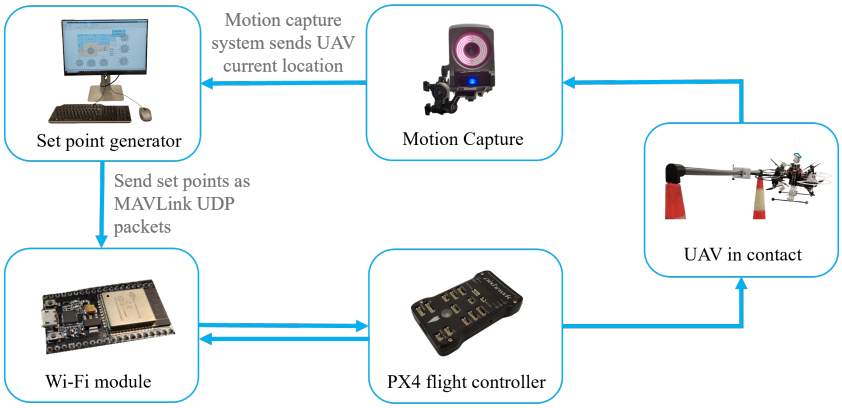


Figure 5: Connection between different parts of the experimental setup

http://www.imavs.org/

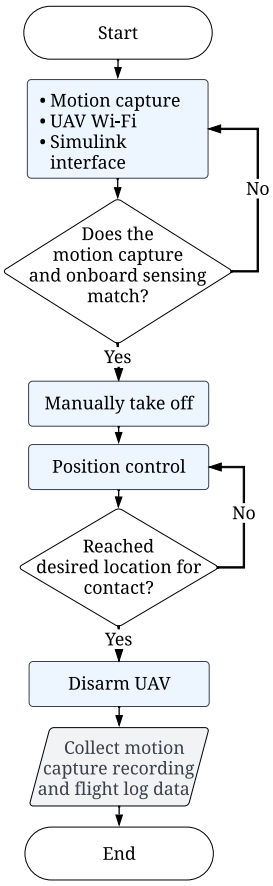


Figure 6: Experimental Process

base gains used in the first in-contact flight experiments with different contact angles.

4 EXPERIMENTAL GAIN TUNING

Table 1 shows the physical contact requirements including the time, force and different angles. Most research studies in UAV contact testing hold contact for 10 seconds [11, 14]. This study initially chose to use 10 seconds; however, it decided to increase this to 30 seconds to showcase the ability of the UAV to sustain prolonged contact. The contact force requirements is 1N or more, which is the 50% of the maximum horizontal force in hover by the utilized UAV in [15]. This force is suitable, as it is within the safe limits of motor saturation for this UAV. Five different angles are chosen to represent a successful and prolonged physical interaction. These angles are chosen, as previous research on contact-based inspection UAVs have decided on these angles as suitable angles required for contact based inspection applications [16]. For each angle there is a coupling between different DoF that conflict. Coupling is when the motion in a certain DoF is linked with the motion in another DoF. Therefore, when the UAV translates or rotates in one DoF, the motion of the other DoF will be affected. This can lead to issues such as a large accumulation of errors in different DoFs. This results in issues such as extreme vibrations, saturation, and instability. Although the UAV is balanced horizontally, due to the end effector attachment, the CoM is slightly above or below the original CoM location. The motor mixer designed and implemented in the firmware for this UAV is optimized for the CoM being precisely in the centre of the UAV. Adding an end effector in angles higher or lower results in the CoM moving. Therefore, due to this, there is an undesired torque affecting the resulting contact for certain angles. In order to eliminate coupling effects, the P-PID gains discussed in Section 3 will be experimented. Although a thorough investigation of gain tuning was conducted with the experimentation of different gain magnitudes and combinations, the minimal amount of change required for successful contact is chosen for the fi-

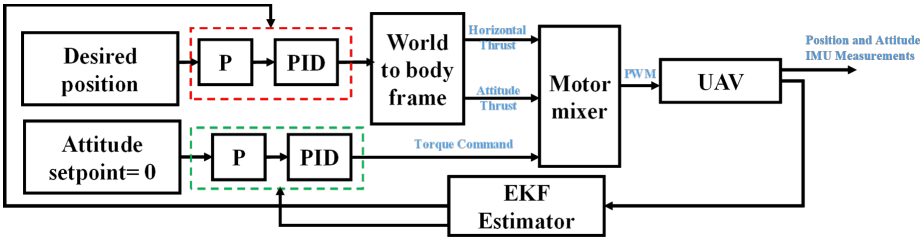


Figure 7: PX4 P-PID Control Architecture

Requirement	Value
Angle selection (deg)	0, 45, 90, -45, -90
Force (N)	≥ 1
Contact time (s)	≥ 30

Table 1: Contact testing requirements

nal experiments. Figure 8 shows all the successful prolonged interaction experiments with different angles[4]. In the following, the gain tuning logic based on the angles and position of CoM are discussed thoroughly:

4.1 0º contact angle

The experiment begins with the end effector and arm inserted into the 0º angle. The first step in this process is to test the 0º angle with the free flight gains. This is to compare later with experiments with modified gains. The 0º contact angle free flight experiments show relatively stable results. Following this, the second experiment repeated the 0º test with gains designed to eliminate coupling effects. For this particular angle, there is a coupling between the X-axis (sideways) and yaw (the axes are shown in Figure 2). There is also a coupling between the Z-axis and the pitch-axis. These couplings have the potential to cause large errors in different axis due to conflicting motions. For example, when the UAV increases in height, if it strays away from the setpoint, it inevitably causes a rotation around the pitch axis, leading to significant pitch errors, integral wind-ups and flips around the pitch axis. Additionally, there is also vibration due to this. Several experiments involve zeroing the axis where less precision is needed for contact for each coupling. Yaw and pitch were zeroed resultantly. These essentially reduce the system's motion from 6 DoF to 4 DoF since there is no longer any control over the pitch and yaw axis. However, for this contact angle, a positive result is seen for either free flight or the modified gains. Vibration is present but not as significant as in other angles, elaborated more below.

4.2 -45º contact angle

Like with the 0º contact angle, it is suspected that coupling would interfere with contact testing at a contact angle of -45º angle, pitch is coupled with Y-axis and Z-axis. Pitch gains are to be zeroed to test if eliminating the coupling will improve contact testing. However, this leads to an unstable flight with large oscillations in the pitch. The free flight gains are better suited to this contact angle over zeroing pitch gains. This is due to the overall UAV CoM being below the original UAV CoM, which is slightly below the mid plane of the stacked rotors due to the end effector weight. This leads to undesired moments, which causes the pitch axis instability; hence, free flight gains are ideal for this contact angle. Although, with free flight gains, there are some minimal vibrations in the pitch axis (due to issues such as coupling between the pitch, Y, and Z axis), it is not significant to prevent stable contact.

4.3 -90º contact angle

Similarly to the 0º angle, it was hypothesized that coupling between axes prevents free flight gains for a stable physical interaction. However, for the -90º contact angle, the axis coupled are different. Here, the pitch is coupled with the Y-axis, and the roll is coupled with the X-axis. Since the X and Y gains are one gain in PX4, pitch gains are zeroed. Like the -45º contact angle, the CoM is below the UAV's original CoM that the motor mixer was created for. Therefore, the undesired moments caused similar oscillations that are observed for -45º. Subsequently, free flight gains result in stable contact with minimal vibrations.

4.4 +45º contact angle

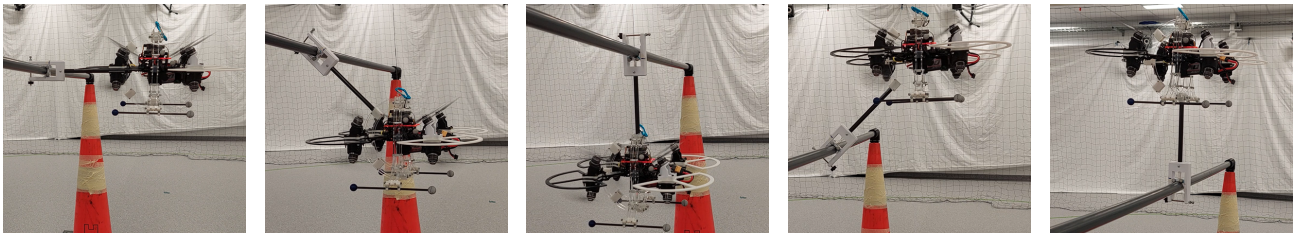
Coupling for this angle is the same as the -45º angle, where the pitch is coupled with the Y and Z axes. However, due to the end effector being angled upwards, the CoM is above the expected location for which the motor mixer was designed. Experiments show that this contact angle is stable with the pitch gains zeroed. Free flight gains will not work for this angle as significant vibration is observed, leading to saturation. Contact is impossible due to the saturation preventing the UAV from applying forces via the end effector on the flexible rod for contact. Further tests demonstrated that only zeroing the pitch gains prevented vibration and saturation issues.

4.5 +90º contact angle

Having a similar coupling with the -90º contact angle, the only reason that makes the gains required different is the lo-

[4]https://youtu.be/4GtdBr-9Ae8

http://www.imavs.org/



(a) 0º (b) 45º (c) 90º (d) -45º (e) -90º

Figure 8: Contact testing for each angle

cation of the COM. Here, the COM is above what the motor mixer expects during experiments, zeroing the pitch gains results in stable flight without any instability observed. This also eliminates excessive vibrations observed with free flight gains.

The above experimental process is repeated several times to validate the results. It is discovered that the pitch axis coupling is the leading cause of vibration and saturation of some rotors. The UAV becomes stable by zeroing the most negative coupling observed in experiments, which is due to the pitch axis, where the UAV tends to vibrate in this axis significantly for certain angles, depending on the position of the CoM. Horizontally, the UAV can always be balanced by shifting the battery back or forth. However, it is difficult to keep the CoM vertically in the UAV's centre if the arm is at angles other than 0º. Table 2 shows the experimental results in a meaningful order. It is observed that if the CoM is above the UAV's central horizontal plane, the pitch gains must be zeroed to avoid vibration from the coupling effect. On the contrary, if the CoM is below the UAV's central horizontal plane, we need to keep the free flight gains, including the pitch gains, as they are to avoid creating an unwanted moment. After tuning the gains, the UAV shows stability such that the overall roll, pitch, and yaw angles remain small. (see Table 3). It can be seen that the UAV is relatively stable with roll, pitch, and yaw varying from the setpoint within $\mp$6º as shown in Figure 9. The compression force for the -45º contact angle is shown in Figure 9a showing the consistent force above 1N for a 30 s time period. Additionally, the attitude over the compression period for this angle are shown in Figure 9b. The mean electrical power values in Table 3 are similar across different angles for comparable compression forces, reflecting the energy required to maintain stable contact. This results from tuning the gains to combat issues such as vibration due to coupling or instability. Before tuning, different DoF are coupled, affecting the contact and requiring a much larger power to make contact with the same force. Table 4 shows that the fully-actuated UAV can successfully maintain a leveled physical contact, which is essential as the arm is rigidly attached to the UAV at a certain angle. The only exception is the 45º contact angle, which has a noticeably larger pitch RMSE of 9.3º, although the response is still stable and meets the time and force requirements.

Angle (deg)	CoM location	Gains
0	Center	Free flight/Zero pitch
45	Above	Zero pitch
90	Above	Zero pitch
-45	Below	Free flight
-90	Below	Free flight

Table 2: Gains required for stable contact

Angle (deg)	Mean force (N)	Mean electrical power (W)
0	2.19	246
45	2.87	396
90	2.42	342
-45	2.90	250
-90	2.54	269

Table 3: Contact testing power and force analysis over 30 second compression period

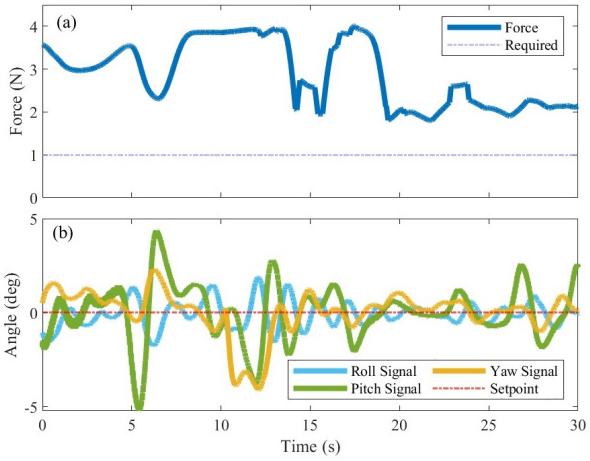


Figure 9: a) Force compression during -45º contact angle. b) Attitude (Euler angles) during physical interaction for -45º

http://www.imavs.org/

Angle (deg)	Roll (deg)	Pitch (deg)	Yaw (deg)
0	0.1	1.4	0.7
45	0.2	9.3	0.6
90	1.6	0.8	0.3
-45	0.7	1.5	1.2
-90	1.2	0.4	0.3

Table 4: RMSE over 30 seconds compression period

5 CONCLUSION

In conclusion, this study delved into the impact of coupling-dependent gain tuning on the prolonged physical interaction of a fully-actuated UAV. Through extensive testing involving five different contact angles for prolonged contact experiments, it became evident that coupling between degrees of freedom can impede sustained contact. The placement of the arm also influences the CoM, introducing complexities, notably when the CoM deviates from the expected 2D plane specified by the motor mixer. This deviation, significantly above the expected plane, results in significant oscillations along the pitch axis, necessitating the zeroing of pitch gains for stability. The study highlights the need for further exploration, suggesting a natural progression to achieve contact tests from free flight gains, flying towards the rod, and switching gains at the point of contact. Exploring different controllers for contact testing across various angles could enhance results. Moreover, research into the potential mid-flight rotation of the manipulator's arm would allow the UAV to take off and rotate the arm to an angle that the UAV cannot take off, e.g., 90º. This would have promising avenues for improved efficiency and versatility in UAV operations.

http://www.imavs.org/

ACKNOWLEDGMENT

The research reported in this article was conducted as part of "Enabling unmanned aerial vehicles (UAVs) to use tools in complex dynamic environments "UOCX2104", which is funded by the New Zealand Ministry of Business, Innovation and Employment.

REFERENCES

[1] Ryan Kitchen, Nick Bierwolf, Sean Harbertson, Brage Platt, Dean Owen, Klaus Griessmann, and Mark A Minor. Design and evaluation of a perching hexacopter drone for energy harvesting from power lines. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1192–1198. IEEE, 2020.

[2] Youngjib Ham, Kevin K Han, Jacob J Lin, and Mani Golparvar-Fard. Visual monitoring of civil infrastructure systems via camera-equipped unmanned aerial vehicles (uavs): a review of related works. *Visualization in Engineering*, 4(1):1–8, 2016.

[3] Julie Linchant, Jonathan Lisein, Jean Semeki, Philippe Lejeune, and Cédric Vermeulen. Are unmanned aircraft systems (uas s) the future of wildlife monitoring? a review of accomplishments and challenges. *Mammal Review*, 45(4):239–252, 2015.

[4] Guangying Jiang, Richard M Voyles, and Jae Jung Choi. Precision fully-actuated uav for visual and physical inspection of structures for nuclear decommissioning and search and rescue. In *2018 IEEE international symposium on safety, security, and rescue robotics (SSRR)*, pages 1–7. IEEE, 2018.

[5] Tzu-Jui Lin and Karl A Stol. Towards automated under-canopy exploration of plantation forests. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1201–1208. IEEE, 2019.

[6] Matko Orsag, Christopher Korpela, Stjepan Bogdan, and Paul Oh. Valve turning using a dual-arm aerial manipulator. In *2014 international conference on unmanned aircraft systems (ICUAS)*, pages 836–841. IEEE, 2014.

[7] Frederico Augugliaro, Sergei Lupashin, Michael Hamer, Cason Male, Markus Hehn, Mark W Mueller, Jan Sebastian Willmann, Fabio Gramazio, Matthias Kohler, and Raffaello D'Andrea. The flight assembled architecture installation: Cooperative construction with flying machines. *IEEE Control Systems Magazine*, 34(4):46–64, 2014.

[8] Pisak Chermprayong, Ketao Zhang, Feng Xiao, and Mirko Kovac. An integrated delta manipulator for aerial repair: A new aerial robotic system. *IEEE Robotics & Automation Magazine*, 26(1):54–66, 2019.

[9] James R Kutia, Karl A Stol, and Weiliang Xu. Aerial manipulator interactions with trees for canopy sampling. *IEEE/ASME transactions on mechatronics*, 23(4):1740–1749, 2018.

[10] Anibal Ollero, Guillermo Heredia, Antonio Franchi, Gianluca Antonelli, Konstantin Kondak, Alberto Sanfeliu, Antidio Viguria, J Ramiro Martinez-de Dios, Francesco Pierri, Juan Cortés, et al. The aeroarms project: Aerial robots with advanced manipulation capabilities for inspection and maintenance. *IEEE Robotics & Automation Magazine*, 25(4):12–23, 2018.

[11] Takahiro Ikeda, Shogo Yasui, Motoharu Fujihara, Kenichi Ohara, Satoshi Ashizawa, Akihiko Ichikawa, Akihisa Okino, Takeo Oomichi, and Toshio Fukuda. Wall contact by octo-rotor uav with one dof manipulator for bridge inspection. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5122–5127. IEEE, 2017.

[12] Teodor Tomić and Sami Haddadin. A unified framework for external wrench estimation, interaction control and collision reflexes for flying robots. In *2014 IEEE/RSJ international conference on intelligent robots and systems*, pages 4197–4204. IEEE, 2014.

[13] Paul E. I. Pounds, Daniel R. Bersak, and Aaron M. Dollar. Grasping from the air: Hovering capture and load stability. In *2011 IEEE International Conference on Robotics and Automation*, pages 2491–2498, 2011.

[14] Karen Bodie, Maximilian Brunner, Michael Pantic, Stefan Walser, Patrick Pfändler, Ueli Angst, Roland Siegwart, and Juan Nieto. Active interaction force control for contact-based inspection with a fully actuated aerial vehicle. *IEEE Transactions on Robotics*, 37(3):709–722, 2021.

[15] Pedro Henrique Mendes Souza. *Dynamics and Control of Unmanned Aerial Vehicles for Prolonged Interaction with Compliant Environments*. PhD thesis, ResearchSpace@ Auckland, 2023.

[16] Robert Watson. *Unmanned Aerial Vehicles for Contact-Based Inspection*. PhD thesis, University of Strathclyde, 2023.

http://www.imavs.org/

Restricted Model Control applied to the Quanser Aero with uncertain parameters

Amélie Carrierou*, Cédric Join †, Jean-Philippe Condomines ‡
∗ and † are with CRAN, Université de Lorraine, BP 239, 54506 Vandœuvre-lès-Nancy, France
‡ is with ENAC, Université de Toulouse, 31400 Toulouse, France

http://www.imavs.org/

ABSTRACT

This work study the impact on a Model-Free Control (MFC) feedback combined with flatness-based control, also know as Restricted-Model Control (RMC), in order to address the problem of robustness to uncertain parameters. The proposed control architecture, is applied to the half-quad Quanser Aero benchmark in two degree-of-freedom configuration. Tests focuse on the robustness of RMC to over-efficient propellers, under-efficient propellers and augmented friction.

1 INTRODUCTION

The number and diversity of applications involving Micro Air Vehicles (MAVs) are extensive and have received considerable attention in recent years. Among possible structures, quadrotor UAVs are a popular option for tasks that require highly agile maneuvers in complex, constrained environments. The Quanser laboratory-based two degree-of-freedom (2-DoF) helicopter system is an ideal platform for such studies due to its versatile design, which allows for the emulation of various flight dynamics and control scenarios. Its ability to operate in both single degree-of-freedom and two degree-of-freedom configurations makes it particularly useful for testing advanced control strategies under different dynamic conditions. Several advanced control techniques like optimal linear control ([1]), sliding mode ([2] and [3]) and adaptive control ([4]) have been employed. However, these approaches require the development of an accurate model describing the dynamics, which is costly and time-consuming.

Flatness-based control (see [5]) offers significant advantages due to its capacity to explicitly derive control inputs from desired trajectories. This approach ensures that complex trajectory planning and execution can be achieved with high accuracy, which is essential for MAVs operating in dynamic and constrained environments. However, real-world applications often involve uncertainties and unmodeled dynamics that can degrade the performance of model-based controllers. For this purpose, Model-Free Control (MFC, see [6]) theory, coupled with flatness-based control, has been developed, providing robustness and enhancing the performance of the control system.

The Quanser Aero, with its configurable propellers and adjustable friction settings, allows for a controlled environment to study the interplay between these two control strategies. By examining the effects of MFC feedback on a flatness-based control scheme, we can better understand how to mitigate the limitations of model-based control in the presence of real-world uncertainties. The experiments conducted on the Quanser Aero will provide valuable insights into the practical implementation of this combined control strategy, ultimately contributing to the development of more reliable and efficient control systems for MAVs.

While recalling the basic motion equations of the Quanser Aero in §II, the main contributions of this paper are therefore:

- to make explicit in §III the theoretical equations that describe RMC architecture in the benchmarking case of the Quanser Aero;
- to provide new preliminary results §IV, focusing on the behaviour of flatness-based control and Restricted-Model-Control with parameters variations.

2 SYSTEM DESCRIPTION

The Quanser Aero is a control benchmark proposed by the Canadian manufacturer Quanser [1]. It's a Multi-Input Muli-Output (MIMO) system with 2 configurations and a direct integration with Matlab-Simulink®. The Quanser Aero, represented Figure 1, have 2 propellers that can be rotated to be either horizontal or vertical. Thus this benchmark have two configurations. On the first one, both propellers are horizontal and this system have only one degree of freedom, pitch rotation. On the second one, one propeller is horizontal and the second one is vertical so this system have 2 degree of freedom, pitch and yaw rotation (see Figure 2). For this work we used the second configuration.

2.1 Modelling

As depicted in the free-body diagram in Figure 2, the horizontal propeller, controlled by the tension V_p, generates a force F_p and induces a torque τ_p, causing angular motion θ in the pitch plane (around the y axis). Similarly, the yaw rotor, driven by the tension V_y, produces an aerodynamic force

*Email address : amelie.carrierou@univ-lorraine.fr
†Email address : cedric.join@univ-lorraine.fr
‡Email address : jean-philippe.condomines@enac.fr

[1] https://www.quanser.com/products/quanser-aero/

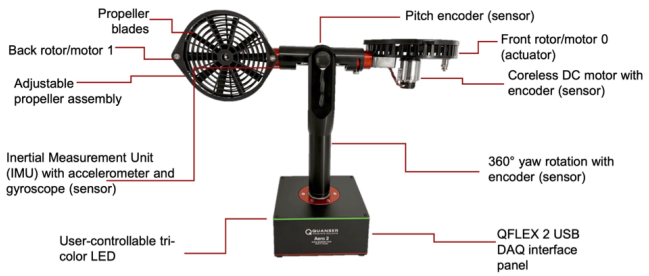


Figure 1: Quanser Aero : 2 degree of freedom configuration [image by Quanser Inc.]

F_y, resulting in torque τ_y and angular motion ψ in the yaw plane (around the z axis). These cross-torques cause significant cross-coupling between pitch and yaw channels. In the Quanser Aero documentation [7], a simple spring-mass-damper type of linear model that takes this coupling into account, is considered. The equations of motion are described in equation (1)

$$\begin{cases} J_p\ddot{\theta} + D_p\dot{\theta} + K_{sp}\theta = \tau_p \\ \\ J_y\ddot{\psi} + D_y\dot{\psi} = \tau_y \end{cases} \tag{1}$$

where τ_p and τ_y are related to their respective motor voltages by

$$\tau_p = K_{pp}V_p + K_{py}V_y, \quad \tau_y = K_{yp}V_p + K_{yy}V_y \tag{2}$$

From the point of view of state-space modelling, the state of this system is $\boldsymbol{x} = [\theta\ \psi\ \dot{\theta}\ \dot{\psi}]^T$, the input is $\boldsymbol{x} = [V_p\ V_y]^T$ and the output is $\boldsymbol{y} = [\theta\ \psi]^T$

Some of the parameters in equation (1) and equation (2) are given in the system's specification, while others were experimentally obtained. The values of the parameters are given by Table 1. In the above description, the term K_{sp} accounts for the gravitational force affecting the vertical motion (i.e, pitch), whereas no such factor is considered for the horizontal motion (i.e, yaw). It is important to note that in this model, the interaction between pitch and yaw movements is captured by the cross-terms K_{py} and K_{yp} with $|K_{py}| = |K_{pp}|$ and $|K_{yy}| = 1.5|K_{yp}|$ (see Table 1), the coupling from V_p to ψ is significantly more than the one from V_y to θ.

2.2 *Settings*

The Quanser Aero model allows for various modifications enabling versatile testing of control strategies and dynamic responses in different experimental scenarios. In this paper, we will concentrate on propeller configurations and friction settings. It is possible to modify the system in two ways to study different aerodynamic and control behaviors. Firstly, the system comes with two main configurations of propellers: a pair with 2 blades and another with 10 blades. Switching between these configurations allows observing how the propeller setup affects the generation of forces and moments, as well as the stability and maneuverability of the system. Secondly, the model offers the option to adjust friction on the horizontal axis by tightening or loosening a screw. This adjustment varies the perceived resistance and inertia by the system, thereby influencing its dynamic response and control accuracy.

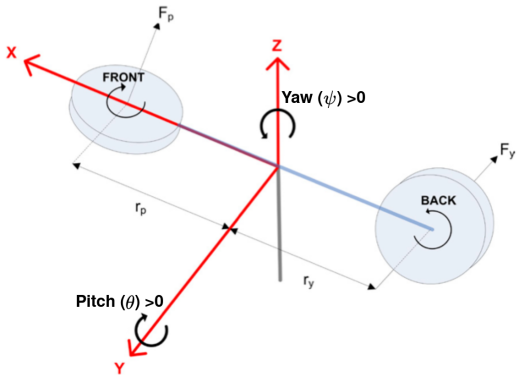


Figure 2: Free-body diagram of the Quanser Aero

Table 1: System parameters

Parameters (10 blades propeller)	Value/range
Moment of inertia about pitch axis ($\mathbf{Jp}$)	0.0219 kgm^2
Moment of inertia about yaw axis($\mathbf{J_y}$)	0.022 kgm^2
Damping about pitch axis ($\mathbf{D_p}$)	0.0113 Vs/rad
Damping about yaw axis ($\mathbf{D_y}$)	0.0131 Vs/rad
Stiffness about pitch axis ($\mathbf{K_{sp}}$)	0.0419 Nm/rad
Torque thrust gain from pitch rotor acting on pitch ($\mathbf{K_{pp}}$)	0.0013 Nm/V
Torque thrust gain from yaw rotor acting on yaw ($\mathbf{K_{yy}}$)	0.002 Nm/V
Torque thrust gain from yaw rotor acting on pitch ($\mathbf{K_{py}}$)	0.0016 Nm/V
Torque thrust gain from pitch rotor acting on yaw ($\mathbf{K_{yp}}$)	-0.0013 Nm/V

3 RESTRICTED MODEL CONTROL

Restricted Model Control (RMC, presented in [6] [8]) represents a compelling synthesis of two control methodologies : flatness-based and Model-Free Control (MFC).

Differential flatness was first introduced by M. Fliess and al. in [5]. It's a structural property of a system that can be used to compute nominal control input based on a dynamical model and a desired output. Flatness based control can successfully solve trajectory tracking problems for complex

http://www.imavs.org/

http://www.imavs.org/

nonlinear systems like for example : induction drive [9], delay systems [10], partial differential equations [11], [12] and tilt-body UAV [13].

Model-Free Control term appears many times in the literature, but in distinct meanings from this paper. Actually, the growing importance of artificial intelligence and machine learning techniques, particularly through neural networks, has naturally been implanted into the model-free terms: see, for example [14] and [15]. However, in this paper, we assume Model-Free Control terms according to [16].

3.1 *Flatness based controller*

Consider a system with state $\boldsymbol{x}(t) \in \mathbb{R}^n$ and input $\boldsymbol{u}(t) \in \mathbb{R}^m$ defined by equation 3.

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{3}$$

This system is said deferentially flat, or just flat, if it exist $\boldsymbol{z}(t) \in \mathbb{R}^m$, called flat output, defined by equation (4) and two function A and B such as the state and the control input and be express based on this flat output.

$$\boldsymbol{z}(t) = \Phi(\boldsymbol{x}(t), \boldsymbol{u}(t), \dot{\boldsymbol{u}}(t), \cdots, \boldsymbol{u}^{(\phi)}(t)) \tag{4}$$

$$\boldsymbol{x}(t) = A(\boldsymbol{z}(t), \dot{\boldsymbol{z}}(t), \cdots, \boldsymbol{z}^{(\alpha)}(t)) \tag{5}$$

$$\boldsymbol{u}(t) = B(\boldsymbol{z}(t), \dot{\boldsymbol{z}}(t), \cdots, \boldsymbol{z}^{(\beta)}(t)) \tag{6}$$

For flatness-based control we need to have a smooth enough reference flat output trajectory $\boldsymbol{z}^*(t)$. Then we use this reference trajectory and equation (6) to compute the nominal input $\boldsymbol{u}^*(t)$. The global input, $\boldsymbol{u}(t)$ is the sum of the nominal input and a feedback term, $\boldsymbol{\delta u}(t)$

$$\boldsymbol{u}(t) = \boldsymbol{u}^*(t) + \boldsymbol{\delta u}(t) \tag{7}$$

Coming back to the Quanser Aero, the output of the system, $\boldsymbol{y}(t) = [\theta(t)\ \psi(t)]^T$ is a flat output. The state vector is trivially obtained with this flat output and its derivative. The relation between the flat output and the input vector is given by equation (8).

$$\begin{cases} V_p = \dfrac{J_p\ddot{\theta} + D_p\dot{\theta} + K_{sp}\theta}{K_{pp} - K_{py}K_{yp}K_{yy}^{-1}} - \dfrac{K_{yp}(J_y\ddot{\psi} + D_y\dot{\psi})}{K_{pp}K_{yy} - K_{py}K_{yp}} \\ V_y = \dfrac{J_y\ddot{\psi} + D_y\dot{\psi} - K_{yp}V_p}{K_{yy}} \end{cases} \tag{8}$$

With this equation and a twice derivable reference trajectory $[\theta^*(t)\ \psi^*(t)]^T$ we have the nominal control input, $[V_p^*(t)\ V_y^*(t)]^T$. The next step is to choose a feedback $[\delta V_p(t)\ \delta V_y(t)]^T$.

With a linear system, one of the simplest choice is to use a feedback proportional to the position error. For the Quanser Aero this gives :

$$V_p = V_p^* + \delta V_p \tag{9}$$

$$V_y = V_y^* + \delta V_y \tag{10}$$

with :

$$\delta V_p = K_1(\theta - \theta^*) + K_2(\psi - \psi^*) \tag{11}$$

$$\delta V_y = K_3(\theta - \theta^*) + K_4(\psi - \psi^*) \tag{12}$$

But it's also possible to construct more a interesting feedback using, for example a data-based control law.

3.2 *Flatness-based model-free control*

Model-Free Control (MFC) is a data-based control architecture proposed by M. Fliess, C. Join and H. Sira-Ramirez ([16]). This controller uses an ultra-local model, given by equation (13), to estimate the unknown dynamic of a system. In this model, ν is the derivative order of y (in general 1 or 2) and α a parameter representing the impact of the input on the dynamic of the output. Both ν and α are chosen by the practitioner. The key step of this theory is the estimation of $F(t)$. It can be made by considering $F(t)$ as a piecewise constant function and using the algebraic estimation technique proposed in [17].

$$y^{(\nu)}(t) = F(t) + \alpha u(t) \tag{13}$$

Once we have $\hat{F}(t)$, the estimate of $F(t)$, the control input is given by equation (14) with $C(e(t))$ a PID controller and $y^*(t)$ a reference output trajectory.

$$u(t) = \frac{-\hat{F}(t) + y^{*(\nu)}(t) + C(e(t))}{\alpha} \tag{14}$$

MFC can be used when there is limited information about the system or when the model is not precise enough. It can also complement model-based controllers to compensate for unmodeled system dynamics.

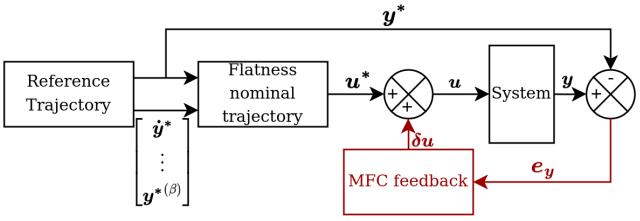


Figure 3: Restricted-Model Controller

The combination of flatness based control and MFC is called Restricted Model Control (RMC). A diagram of this control strategy is proposed figure 3, (to keep this diagram clear, we illustrated the case where the output of the system is also a flat output). In the RMC architecture, the ultra-local model given by equation (13) is adapted to represent the dynamic of the error $e_y = y - y^*$ thus we have :

$$e_y^{(\nu)}(t) = F(t) + \alpha\delta u(t) \tag{15}$$

From this model, equation (14) is rewrited stabilised the error in 0.

$$\delta u(t) = \frac{-\hat{F}(t) + C(e(t))}{\alpha} \tag{16}$$

For the Quanser Aero, the ultra-local model have to be adapted for a MIMO systems. Extension of MCF to this kind of systems where used in [18] and [19]. Since the cross coupling gains, K_{py} and K_{yp}, are important, we choose to use coupled ultra-local model. By taking the second derivative of the position errors, we have an ultra-local model given by :

$$\begin{cases} \ddot{e}_\theta(t) = F_\theta(t) + \alpha_{pp}\delta V_p(t) + \alpha_{py}\delta V_y(t) \\ \ddot{e}_\psi(t) = F_\psi(t) + \alpha_{yp}\delta V_p(t) + \alpha_{yy}\delta V_y(t) \end{cases} \tag{17}$$

Using the previously mentioned algebraic technique, an estimation of F_θ can be :

$$\hat{F}_\theta(t) = \frac{60}{T_{f\theta}^5}\int_0^{T_{f\theta}} \left((T_{f\theta}-\tau)^2 - 4\tau(T_{f\theta}-\tau) + \tau^2\right)\tilde{e}_\theta(\tau) - \frac{\alpha_{pp}}{2}\tau^2(T_{f\theta}-\tau)^2\delta\tilde{V}_p(\tau) - \frac{\alpha_{py}}{2}\tau^2(T_{f\theta}-\tau)^2\delta\tilde{V}_y(\tau)d\tau \tag{18}$$

with $T_{f\theta}$ the time window, $\tilde{e}_\theta(\tau) = e_\theta(\tau+t-T_{f\theta})$, $\delta\tilde{V}_p(\tau) = \delta V_p(\tau + t - T_{f\theta})$ et $\delta\tilde{V}_p(\tau) = \delta V_y(\tau + t - T_{f\theta})$. $\hat{F}_\psi$ is computed by replacing $T_{f\theta}$ by $T_{f\psi}$, $\tilde{e}_\theta(\tau)$ by $\tilde{e}_\psi(\tau)$, α_{pp} by α_{yp} and α_{py} by α_{yy} .

The quality of this estimation depend on the choice of $T_{f\theta}$. To have a good estimate, $T_{f\theta}$ have to bee small enough but a larger $T_{f\theta}$ guarantee better robustness to measurement noise.

With the two equations of the system (17) we can obtain the expression of $\delta V_y(t)$ given by equation (19). Then, with the first equation of this system, we have the expression for $\delta V_p(t)$ (equation (20)). Once we have those equations, we can use $\hat{F}_\theta(t)$ and $\hat{F}_\psi(t)$ to compute the values of $\delta V_y(t)$ and $\delta V_p(t)$.

$$\delta V_y(t) = \frac{-\hat{F}_\psi(t) + C(e_\psi(t))}{\alpha_{yy} - \alpha_{py}\alpha_{yp}\alpha_{pp}^{-1}} - \frac{\alpha_{yp}(\hat{F}_\theta - C(e_\theta(t))}{\alpha_{pp}\alpha_{yy} - \alpha_{py}\alpha_{yp}} \tag{19}$$

$$\delta V_p(t) = \frac{-\hat{F}_\theta(t) - \alpha_{py}\delta V_y + C(e_\theta(t))}{\alpha_{pp}} \tag{20}$$

For $C(e_\psi(t))$ and $C(e_\theta(t))$ we have chosen a proportional derivative controller :

$$C(e(t)) = k_p e(t) + k_d\dot{e}(t) \tag{21}$$

With a desired double pole s_d the k_p and k_d gains are :

$$k_p = -s_d^2 \tag{22}$$

$$k_d = -2s_d \tag{23}$$

Starting from a flatness-based controller, RMC require to chose a order of derivation, ν, the α parameter and the time window T_f used for estimation. Those parameters are adjusted based on trials and fails. The choice of α can be guided by the model and the choice of T_f depends on the dynamic of the systems and the measurement noise. Compared to the research of a flat input and the computation of nominal trajectory, the tuning of RMC parameters does not require a lot of effort but can improve the flatness-based control, especially in the case of partially know dynamics or imprecise model.

4 TESTS AND RESULTS

We now apply the control approach described in the previous section for Quanser Aero whose specifications are described in Table I. The idea is to investigate the impact of MFC feedback on flatness-based control in terms of uncertain parameters.

To investigate the impact of MFC feedback on flatness-based control, we leveraged the capability to modify the behavior of the Quanser Aero. The objective of the tests was to track a step signal in pitch while maintaining yaw at 0 degrees. Both control laws require the reference signal to be at least twice differentiable, thus the step signal was filtered by a discreet second-order system. The relation between θ_{ref} (the reference) and θ^* (filtered reference) and its first and second derivative are given by equation (24) to (26) with T_e the sampling time of the Quanser Aero.

$$\theta^*(t) = \frac{\theta_{ref}(t) + (2*\omega + 2*\omega^2)*\theta^*(t-T_e) - \omega^2*\theta^*(t_2T_e)}{\omega^2 + 2\omega + 1} \tag{24}$$

$$\dot{\theta}^*(t) = \frac{\theta_{ref}(t) - \theta^*(t-T_2)}{T_e} \tag{25}$$

$$\ddot{\theta}^*(t) = \frac{\theta_{ref}(t) - 2\theta^*(t-T_e) + \theta^*(t-2T_e)}{T_e^2} \tag{26}$$

The ω parameter is used to tune the filter. With $\omega = 300$ the input (θ_{ref}) and filtered input (θ^*) are presented figure 4.

to maximise the cases studied, the tests have been dived in two categories :

1. Tests with a 10 blades nominal system (Figures 5 to 8)
2. Tests with a 2 blades nominal system (Figures 9 to 12)

In both cases the nominal trajectory computation and controllers tuning where made on the nominal system then used without any modification on the altered system. One the first case (10 blades nominal system) we have studied the effect of an under-efficient propellers and augmented friction. On

http://www.imavs.org/

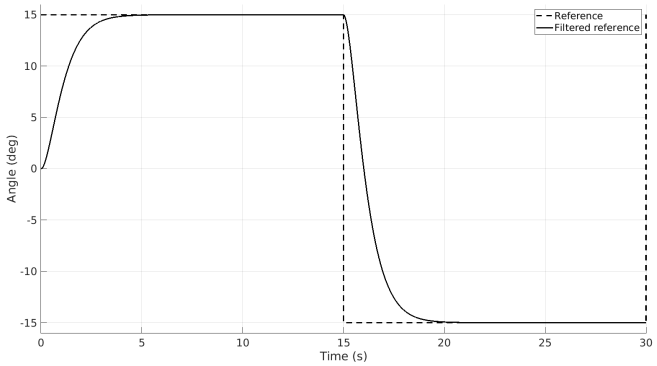


Figure 4: Reference and filtered reference

the second one, the effects of over-efficient propellers where tested.

The tuning of the two controllers for those both cases are presented table 2. Since the objectives and dynamics are different in pitch and yaw regulation thus two different sizes of time window are used. We have $T_{f\theta} = 0.1$s and $T_{f\psi} = 0.5$s. We also used the same α parameters for the two RMC controller : $\alpha_{pp} = 0.35$, $\alpha_{py} = 0.02$, $\alpha_{yp} = -0.02$ and $\alpha_{yy} = 1$.

Table 2: Values of gains tuning

Gains	10 blades system	2 blades system
K_1	-200	-480
K_2	90	80
K_3	-200	-480
K_4	-90	-80
s_θ	3	3.8
s_ψ	4.1	7.5

The gains presented in the table above where chosen to ensure the best pitch trajectory following while trying to maintain a reasonable disturbance rejection on yaw.

Since the feedback term for the flatness based controller is only proportional to positions errors, the steady state error in nominal cases was expected. Our objective is to study the impact of parameters variation on both control laws, not there trajectory tracking performances.

As shown in Figure 6 under-efficient propellers (yellow line) increased the steady state error compared to nominal case (red line). In the first 15 seconds the value of the error was increased by 13% and by 17% in the last 15 seconds. Restricted-Model Control (Figure 5) achieved to nullify this steady-state error, with nominal and under-efficient propellers, while preserving a good behaviour in transient response. This better response in pitch come at the prise of a poorer disturbance rejection in yaw (Figure 7 for RMC and 8 for flatness-based control).

With increased friction (green line), flatness-based control have the same kind of result with a more important effect on transient response but a lower steady-state error on the lasts 15 seconds (11%). Like on the previous test, the RMC converge at the same pitch value than on the nominal case, with some small oscillations on the transient response. To follow the pitch trajectory despite this modification, the RMC generate a higher control input, which leads to greater perturbation on yaw (with pics value of $+16$ degree and -13.5 degree for RMC, see Figure 5). The same effect is observed on flatness-based regulation but with much lower pics value.

Our last tests where made on with a 2 blades nominal system to tests the effect of over efficient propellers. Since the propeller is less effective, the reference amplitude for pitch was divided by two. With this lower amplitude, the coupling effect is less important so the yaw response is great for both controllers (error between -1 and 1 degree for RMC and -0.5 and 0.9 degree for flatness-based controller). For pitch trajectory following, the controller have better performances in transient and steady-state response. But on the flatness-based controller response, especially in figure 12, we can see some regular oscillations on the yellow curve. With higher feedback gains values those oscillations had lead to instability. We did not had this problem with RMC. With this lower reference output value we also had difficulties with the high quantization step size of the Quanser Aero.

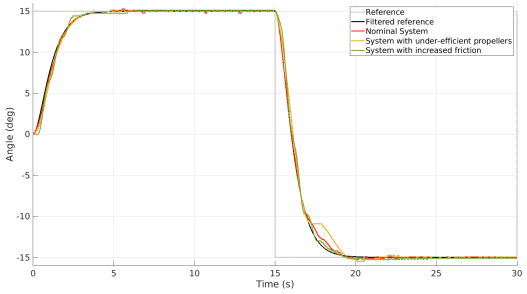


Figure 5: Pitch response, RMC, 10 blades nominal system

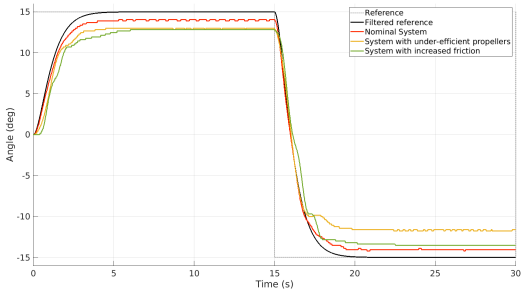


Figure 6: Pitch response, flatness controller, 10 blades nominal system

In all our tests, RMC has shown more robustness to parameters variation in the pitch response but at the price of a poorer yaw response. This is show that the MFC feedback

http://www.imavs.org/

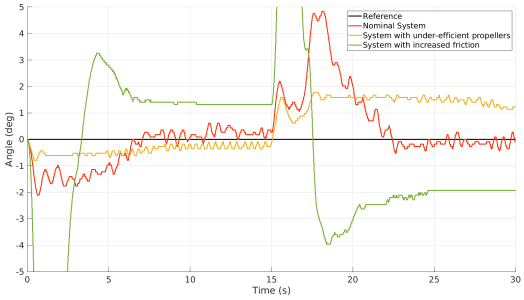


Figure 7: Yaw response, RMC, 10 blades nominal system

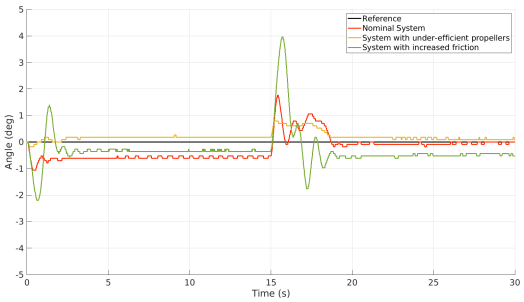


Figure 8: Yaw response, flatness controller, 10 blades nominal system

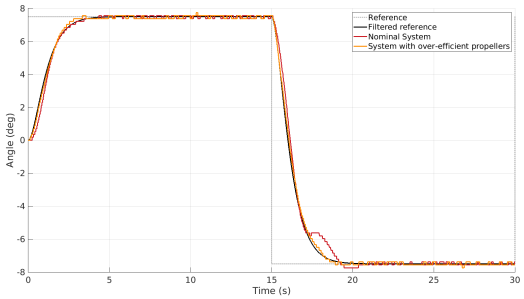


Figure 9: Pitch response, RMC, 2 blades nominal system

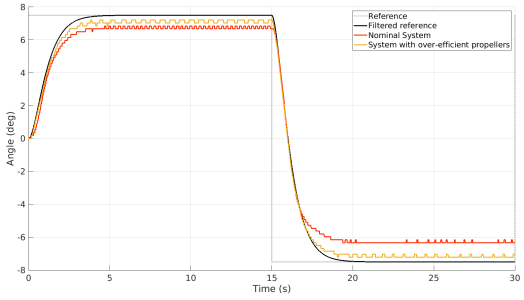


Figure 10: Pitch response, flatness controller, 2 blades nominal system

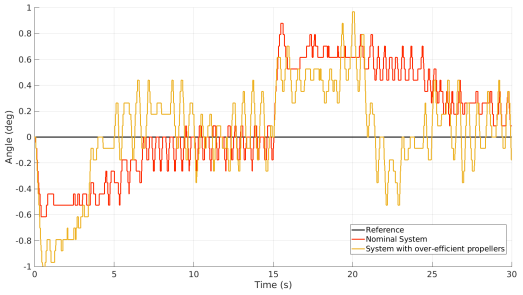


Figure 11: Yaw response, RMC, 2 blades nominal system

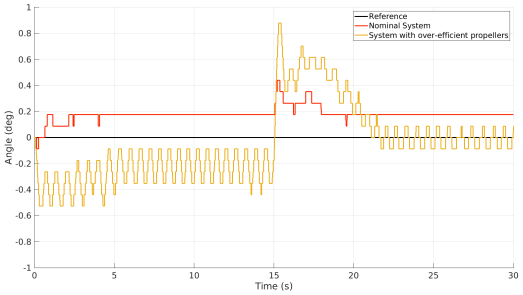


Figure 12: Yaw response, flatness controller, 2 blades nominal system

can improve a flatness based control law in when the model is partially known.

5 CONCLUSION

In this paper, we have design a MIMO Restricted-Model Controller for the 2 DoF Quanser Half-quad in order to address the problem of uncertain parameters identification. The proposed design was test on several cases and compared with flatness-based control. Those tests have shown that Model-Free Control can be used to improve the adaptive properties of flatness-based control.

This preliminary study has produced encouraging results and this control strategy will be applied on quadrotor MAV on future work.

REFERENCES

[1] Falguni Gopmandal and Arun Ghosh. Lqr-based mimo pid control of a 2-dof helicopter system with uncertain cross-coupled gain. *IFAC-PapersOnLine*, 55(22):183–188, 2022.

[2] Peter Lambert and Mahmut Reyhanoglu. Observer-based sliding mode control of a 2-dof helicopter system. In *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*, pages 2596–2600. IEEE, 2018.

[3] Sami Labdai, Larbi Chrifi-Alaoui, Saïd Drid, Laurent Delahoche, and Pascal Bussy. Real-time implementa-

http://www.imavs.org/

tion of an optimized fractional sliding mode controller on the quanser-aero helicopter. In *2020 International Conference on Control, Automation and Diagnosis (ICCAD)*, pages 1–6. IEEE, 2020.

[4] Fouad Yacef, Nesrine Benkhaled, Lina Benhammouda, Lamia Melkou, Laid Degaa, Nassim Rizoug, and Mahmoud Belhocine. Adaptive intelligent control of 2-dof helicopter system. In *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 2734–2739. IEEE, 2023.

[5] M Fliess, J Lévine, P Martin, and P Rouchon. On differentially flat nonlinear systems. *IFAC Proceedings Volumes*, 25(13):159–163, 1992.

[6] Michel Fliess and Cédric Join. Model-free control. *International journal of control*, 86(12):2228–2252, 2013.

[7] Quanser. Quanser aero 2 dof lab guide, 2016.

[8] Jorge Villagra and David Herrero-Pérez. A comparison of control techniques for robust docking maneuvers of an agv. *IEEE Transactions on Control Systems Technology*, 20(4):1116–1123, 2011.

[9] Veit Hagenmeyer and Emmanuel Delaleau. Continuous-time non-linear flatness-based predictive control: an exact feedforward linearisation setting with an induction drive example. *International Journal of Control*, 81(10):1645–1663, 2008.

[10] Maria Bekcheva, Hugues Mounier, and Luca Greco. Control of differentially flat linear delay systems with constraints. *IFAC-PapersOnLine*, 50(1):13348–13353, 2017.

[11] AF Lynch and J Rudolph. Flatness-based boundary control of a class of quasilinear parabolic distributed parameter systems. *International Journal of Control*, 75(15):1219–1230, 2002.

[12] Birgit Schörkhuber, Thomas Meurer, and Ansgar Jüngel. Flatness of semilinear parabolic pdes—a generalized cauchy–kowalevski approach. *IEEE Transactions on Automatic Control*, 58(9):2277–2291, 2013.

[13] Ezra Tal and Sertac Karaman. Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness. *IEEE Transactions on Control Systems Technology*, 29(3):1203–1218, 2020.

[14] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[15] Mircea-Bogdan Radac, Radu-Emil Precup, and Raul-Cristian Roman. Model-free control performance improvement using virtual reference feedback tuning and reinforcement q-learning. *International Journal of Systems Science*, 48(5):1071–1083, 2017.

[16] Michel Fliess, Cédric Join, and Hebertt Sira-Ramirez. Complex continuous nonlinear systems: their black box identification and their control. *IFAC Proceedings Volumes*, 39(1):416–421, 2006.

[17] Michel Fliess and Hebertt Sira-Ramirez. An algebraic framework for linear identification. *ESAIM: Control, Optimisation and Calculus of Variations*, 9:151–168, 2003.

[18] Frédéric Lafont, Jean-François Balmat, Nathalie Pessel, and Michel Fliess. A model-free control strategy for an experimental greenhouse with an application to fault accommodation. *Computers and Electronics in Agriculture*, 110:139–149, 2015.

[19] Ouassim Bara, Michel Fliess, Cédric Join, Judy Day, and Seddik M Djouadi. Toward a model-free feedback control synthesis for treating acute inflammation. *Journal of theoretical biology*, 448:26–37, 2018.

http://www.imavs.org/

On the Lateral Non-Stabilizability of PID Controllers for Vision-Based Line-Following Multicopters through Roll Angle Setpoints

P. Van Holm, A. B. Mallikarjuna, C. Simon Soria and L. R. Lustosa *
ISAE-SUPAERO, Université de Toulouse, 31400 Toulouse, France

http://www.imavs.org/

ABSTRACT

This paper investigates challenges in designing a single-input, single-output stabilizing controller for the lateral dynamics of vision-based autonomous line-following multicopters using roll angle setpoints. The system input is at the multicopter roll angle setpoint level. At the same time, its output is the outcome of a computer vision system that yields the distance of the line marker reference to the optical center of the image frame in pixels. After linearization, we show that the resultant transfer function is nonminimum phase. Furthermore, we show that pure Proportional-Integral-Derivative (PID) controllers render the closed-loop plant unstable for all gains values. To solve this issue, we propose an additional feedforward term, prove its effectiveness in theory, and validate it through multicopter simulations including multicopter dynamics with its internal cascaded stabilizing angle and angular rate loops. Furthermore, we present a computer vision algorithm for line tracking using contour analysis and Hue-Saturation-Value color segmentation. Lastly, we validate our controller in fully integrated system through flight tests in a realistic simulation of an indoors environment.

1 INTRODUCTION

This work studies the autonomous trajectory tracking problem of a multicopter drone using downwards-facing camera measurements of a visual line reference on the ground. Typical applications and scenarios include power transmission lines and pipeline following and inspection. While the commonplace strategy for solving this problem involves a guidance law that sets yaw angle setpoints to the flight management unit (FMU) — responsible for implementing low-level flight stabilization through cascaded angle and angular velocity proportional-integral-derivative (PID) controllers —, we examine in this work an alternative solution involving roll angle setpoints instead. However, since rolling towards a line forcibly casts the line even away in the image plane, this control strategy yields a nonminimal phase system. We prove this fact through linearization and study of the resulting transfer function. Furthermore, we show that no PID controller gains would stabilize the closed-loop system. This fact might explain why this strategy is not found in the literature.

1.1 Motivation

Our interest in line-following multicopters rose originally from IMAV 2024's indoor competition description [1], and is driven by the applications that can be derived from it. As explained in [2, 3], the wide application spectrum covers areas such as monitoring and surveillance, infrastructure inspection, precision agriculture, or real-time monitoring of road traffic. Inspection of power lines and pipelines [4, 5, 2] together with crop row detection [6] are among the most discussed topics regarding autonomous UAV's that require line-following capabilities.

Additionally, the interest of rejecting GNSS positioning and navigation follows from the unfeasibility of flying GNSS-reliant drones in deprived areas such as indoors or large cities [7]. These systems rely on external data, making them susceptible to hijacking or spoofing [8, 9, 3] and are expensive in terms of weight and power budget [7]. GNSS-reliant UAV's do not perform well at low altitudes either, and as [10] indicated in 2015, "Small autonomous drones flying at low altitude will need more complex levels of control autonomy and additional sensors to detect distances from the surrounding environment and perform safe and stable trajectories. Vision is a promising sensor modality for small drones".

1.2 Related work

Most previous studies deliberately avoid combining vision-based localization techniques and roll control. In [11, 4, 5, 6, 12, 13], the multicopter's orientation is adjusted using only yaw-angle inputs. In [14], the guidance law similarly sets the yaw-angle setpoints through a spatially discretized control action conditioned on the line's location inside the image. Another strategy employed by [15] is the use of an inward-tilted rotor configuration that allows for lateral movement with reduced induced roll or pitch angles. This strategy enabled the authors to minimize the distance to the line without significant roll, thereby avoiding the non-

*Email addresses: pieter.van-holm@student.isae-supaero.fr, apoorva.busunur-mallikarjuna@student.isae-supaero.fr, carmelo.simon-soria@student.isae-supaero.fr, leandro.ribeiro-lustosa@isae-supaero.fr

http://www.imavs.org/

minimal phase system considered in this paper. To completely avoid the issue, the authors in [16, 17, 18] use an inertially stabilized gimbal camera, which keeps the camera's orientation aligned with the ground and independent of the roll, pitch, and yaw motion of the multicopter.

In contrast to these methods, the approach proposed by the authors in [19] for their autonomously landing multicopter aligns conceptually with the idea presented in this paper. In their work, the distance to the landing target is estimated using a downward-facing camera, which is then employed to control the quadcopter's XY-position via PID roll and pitch controllers. Similar to our approach, they account for the additional distance introduced by the quadcopter's tilt. However, in their method, this control strategy is integrated into an outer control loop, while an inner control loop utilizes gyros and optical flow to stabilize the quadcopter and maintain proximity to the target. Furthermore, a GNSS-dependent position controller is used beforehand to bring the quadcopter close to the target. The camera-based distance measurements are thus not used as the sole inputs for completely autonomous navigation, thereby ignoring the challenge this introduces. In contrast, our method relies exclusively on visual input for controlling the multicopter's positioning, offering a more direct approach to autonomous vision-based navigation and control.

In general, prioritizing roll-angle setpoints is beneficial, as rolling dynamics are typically faster and less prone to saturation. Notably, although this work focuses on the multicopter case, the problem is analogous to fixed-wing architectures, which similarly rely on roll maneuvers to execute turns.

1.3 Paper structure

This paper is structured as follows: Section 2 states the problem and the desired control structure architecture. Section 3 contains the main contributions of this paper and shows that although no PID controller gains stabilize the proposed system, a feedforward term exists that renders the unstable PID controllers closed-loop stable for a nonempty set of gains. Sections 4 and 5 illustrate a practical hardware/software implementation for the required computer vision algorithms and multicopter architecture for validating the theory. Section 6 explains the practical implementation and shows the results of flight simulations using the proposed roll setpoint controller with feedforward compensation. Section 7 closes the paper with the conclusion and discusses future work.

2 PROBLEM STATEMENT

2.1 Dynamics model

The main focus of this work will be the control of the multicopter's side motion through roll. A simplified 2D model is considered, consisting of a fixed reference frame (Y_L, Z_L) aligned with the line, and a moving reference frame (y_c, z_c) aligned with the camera on the bottom of the multicopter. Figure 1 displays the system and the difference vector $\boldsymbol{p_{l/d}} = (0, D, -h)^T$ between the two reference frames. The difference vector considered in the fixed reference frame is noted $\boldsymbol{p^L_{l/d}}$ and in the camera's reference frame $\boldsymbol{p^c_{l/d}}$.

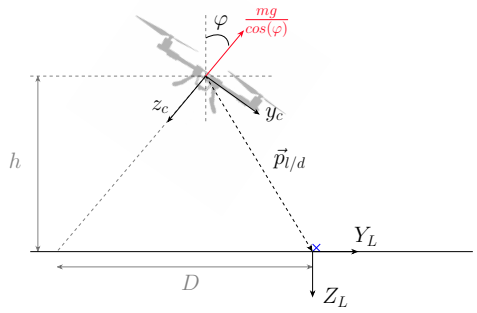


Figure 1: **2-Dimensional configuration:** Two-dimensional representation of the multicopter and the line.

For the camera, a simple pinhole model is considered. The distance d seen by the camera in pixels, as shown in Figure 2, corresponds to the real distance D according to the relation

$$\frac{d}{f} = \frac{D}{h} \Leftrightarrow d = f \cdot \frac{D}{h}. \tag{1}$$

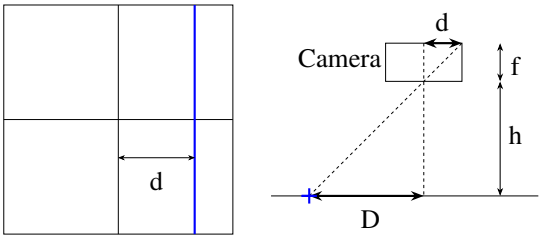


Figure 2: **Pinhole Camera model:** The distance to the line seen on the screen compared to the real the real distance to the line.

2.2 State space model derivation

A two-dimensional state vector $\boldsymbol{x} = (p_y\ v_y)^T$ composed of the position and velocity of the camera along the y-axis is chosen. Let the output of the system y be the measured distance d on the camera frame in pixels and the control input u the multicopter's roll angle φ. Assuming the multicopter will only undergo small angle variations, the derivative of the state vector is obtained:

$$\dot{\vec{x}} = \frac{d}{dt}\begin{pmatrix} p_y \\ v_y \end{pmatrix} = \begin{pmatrix} v_y \\ g \cdot \tan(\varphi) \end{pmatrix} \approx \begin{pmatrix} v_y \\ g \cdot \varphi \end{pmatrix}. \tag{2}$$

To express the measured distance d in the camera's frame, the vector $\boldsymbol{p^L_{l/d}}$ must be rotated around the x-axis according

http://www.imavs.org/

to the rotation matrix $\boldsymbol{R_L^c}$:

$$\vec{p}_{l/d}^{\,c} \equiv \begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix} = R_L^c \cdot \vec{p}_{l/d}^{\,L} = \begin{pmatrix} 0 \\ h\sin(\varphi) - D\cos(\varphi) \\ h\cos(\varphi) + D\sin(\varphi) \end{pmatrix}.$$

Knowing $\boldsymbol{p_{l/d}^c}$, Equation (1) is expressed in the camera's reference frame:

$$d = f \cdot \frac{d_y}{d_z} = f \cdot \frac{h\sin(\varphi) - D\cos(\varphi)}{h\cos(\varphi) + D\sin(\varphi)}. \tag{3}$$

Calculating the partial derivatives and evaluating at the point $(0, 0)$ yields

$$\begin{cases} \dot{\vec{x}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{bmatrix} 0 \\ g \end{bmatrix} u \\ \\ y = \begin{bmatrix} -\frac{f}{h} & 0 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{bmatrix} f \end{bmatrix} u \end{cases}. \tag{4}$$

The controllability and observability matrices

$$\mathcal{O} = \begin{bmatrix} 0 & g \\ g & 0 \end{bmatrix} \; ; \; \mathcal{K} = \begin{bmatrix} -f & 0 \\ 0 & -f \end{bmatrix} \tag{5}$$

are full rank, showing that the system is both controllable and observable. The resulting transfer function is

$$\begin{aligned} G(s) &= C \cdot (s \cdot I - A)^{-1} \cdot B + D \\ &= f \cdot \left(\frac{s^2 - \frac{g}{h}}{s^2} \right). \end{aligned} \tag{6}$$

Equation (6) shows the presence of a zero in the right half of the s-plane, indicating a non-minimum phase system. To reduce the distance to the line d, the multicopter must roll in the direction of the line, creating the acceleration $a_y = g \cdot \tan(\varphi)$. This maneuver temporarily increases the error d, exemplifying the well-known characteristic behavior of non-minimum phase systems. Unlike many typical and counter-intuitive examples used to illustrate non-minimum phase behavior, the dynamics of this system is easily understood and visualized, making it particularly valuable for educational purposes.

3 CONTROL OF THE SYSTEM

This section outlines the controller design aimed at achieving the desired system behavior. Initially, the system's instability under a PID controller is demonstrated. Subsequently, a feed-forward mechanism is introduced to stabilize the system. The section concludes with simulation results, showcasing the control performance.

3.1 Instability under PID control

Using a PID controller of the form

$$C(s) = K_p + K_I \frac{1}{s} + K_d s$$

in a feedback control loop yields the following closed-loop transfer function:

$$\begin{aligned} H_{cl}(s) &= \frac{C(s)G(s)}{1 + C(s)G(s)} \\ &= \frac{s^4 + \left(\frac{K_p}{K_d}\right) s^3 + \left(\frac{K_I}{K_d} - \frac{g}{h}\right) s^2 - \left(\frac{K_p}{K_d}\frac{g}{h}\right) s - \left(\frac{K_I}{K_d}\frac{g}{h}\right)}{s^4 + \left(\frac{fK_p+1}{fK_d}\right) s^3 + \left(\frac{K_I}{K_d} - \frac{g}{h}\right) s^2 - \left(\frac{K_p}{K_d}\frac{g}{h}\right) s - \left(\frac{K_I}{K_d}\frac{g}{h}\right)}. \end{aligned} \tag{7}$$

A necessary condition for stability is that the coefficients of the polynomial in the denominator must be non-negative. This leads to the constraints

$$\begin{cases} \frac{K_p+1/f}{K_d} > 0 \\ \frac{K_I}{K_d} > \frac{g}{h} \\ \frac{K_p}{K_d} < 0 \\ \frac{K_I}{K_d} < 0 \end{cases}. \tag{8}$$

Considering the case where $K_d > 0$, the combination of the second and last constraint of Equation (8) lead to an empty set for the possible values K_I can take. Considering the second case where $K_d < 0$, the directions of the inequalities are flipped and a similar contradiction is obtained. As a result, a PID controller is not able to stabilize the system in a feedback control loop.

3.2 Feed-forward compensation combined with a PD controller.

Due to the inability of the PID controller to stabilize the system, an alternative control strategy is investigated. In particular, a feed-forward compensation combined with a PD controller is introduced.

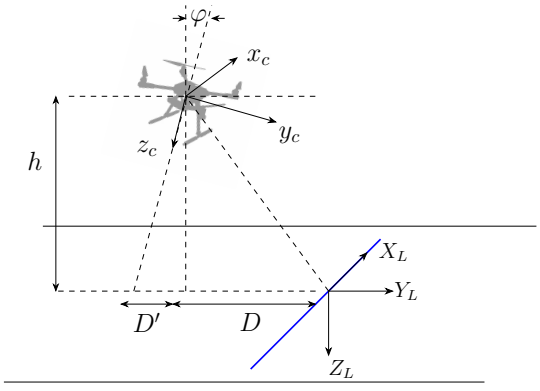


Figure 3: **3-Dimensional configuration:** Visualization of the additional error D' created by the rolling of the multicopter.

A measurement h_2 of the multicopter's height h_1 is used and the additional error D' created by the roll angle, as illustrated in Figure 3, can be fed forward. The additional error

is approximated as $D' = h\tan(\varphi) \approx h\varphi$, making again the assumption of small angles. By subtracting D' to the output of the system, the remaining error will be the real distance D between the multicopter and the line. The resulting control scheme is shown in Figure 4.

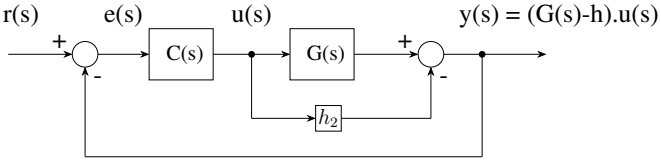


Figure 4: **Control scheme:** Addition of the feed-forward term to compensate the error.

By considering a PD controller

$$C(s) = K_p + K_d s\,, \tag{9}$$

the closed-loop transfer function becomes

$$\begin{aligned} H_{cl}(s) &= \frac{C(s)(G(s)-h_2)}{1+C(s)(G(s)-h_2)} \\ &= \frac{s^3 + \left(\frac{K_p}{K_d}\right)s^2 - \left(\frac{fg}{h_1(f-h_2)}\right)s - \left(\frac{K_p}{K_d}\frac{fg}{h_1(f-h_2)}\right)}{s^3 + \left(\frac{K_p}{K_d} + \frac{1}{K_d(f-h_2)}\right)s^2 - \left(\frac{fg}{h_1(f-h_2)}\right)s - \left(\frac{K_p}{K_d}\frac{fg}{h_1(f-h_2)}\right)}\,, \end{aligned} \tag{10}$$

where a distinction is made between the actual height of the multicopter h_1 present in the system Equation (6) and the measured height h_2 to emphasise that these are different values.

The Routh-Hurwitz criterion for a third order transfer function $d(s) = s^3 + a_2 s^2 + a_1 s + a_0$ ensures that the closed-loop system is stable if and only if the following two conditions are met: $a_2, a_1, a_0 > 0$ and $a_2 \cdot a_1 > a_0$. The transfer function (10) is rewritten to

$$H_{cl} = \frac{s^3 + \lambda s^2 - \gamma s - \lambda\gamma}{s^3 + (\lambda+\alpha)s^2 - \gamma s - \lambda\gamma} \tag{11}$$

by defining

$$\lambda = \frac{K_p}{K_d}\,;\ \gamma = \frac{fg}{h_1(f-h_2)}\,;\ \alpha = \frac{1}{K_d(f-h_2)}. \tag{12}$$

Consequently, the first Routh-Hurwitz condition yields the constraints:

$$\begin{cases} \lambda+\alpha > 0 \\ \gamma < 0 \\ \lambda\gamma < 0 \end{cases} \Leftrightarrow \begin{cases} \lambda > -\alpha \\ \gamma < 0 \\ \lambda > 0 \end{cases}. \tag{13}$$

The second constraint is met when $h_2 >> f$ is considered. The hovering height of the multicopter must be greater than the camera's focal length, which is the case for real systems. The second Routh-Hurwitz condition yields:

$$(\lambda+\alpha)(-\gamma) > (-\lambda\gamma) \Leftrightarrow \alpha > 0\,, \tag{14}$$

which enforces that K_d must be negative. Consequently, the third constraint of Equation (13) dictates that K_p must also be negative. The first constraint of Equation (13) yields

$$K_p < \frac{1}{h_2 - f}\,, \tag{15}$$

which is always met when $h_2 >> f$ and $K_d < 0$. As a result, the addition of the feed-forward term allows the stabilization of the system using the PD controller in Equation (9).

The robustness of to feedback loop to variations of the measurement h_2 must be ensured. Considering small variations δh_1 such that $h_2 = h_1 + \delta h_1$, Equation (12) changes to

$$\lambda = \frac{K_p}{K_d}\,;\ \gamma = \frac{fg}{h_1(f-h_1-\delta h_1)}\,;\ \alpha = \frac{1}{K_d(f-h_1-\delta h_1)}\,. \tag{16}$$

The previously obtained conditions (13) will be violated when $\delta h_1 < 0$ and $|\delta h_1| > |h_1|$, which goes against the assumption that δh_1 is a small variation.

3.3 Simulation

A Matlab Simulink model incorporating the nonlinear dynamics of a quadrotor and the Pixhawk attitude controller is implemented as a first proof of concept for the proposed controller. To model realistic operational constraints, a roll angle limit of 0.26 radians ($\approx 15°$) has been imposed on the controller output. This constraint ensures that the roll angle remains within realistic bounds during simulation.

The system's performance in tracking a line with abrupt changes is evaluated. The results from the simulations, illustrated in Figure 5, indicate that roll adjustments allow the multicopter to follow the line and react to the abrupt changes.

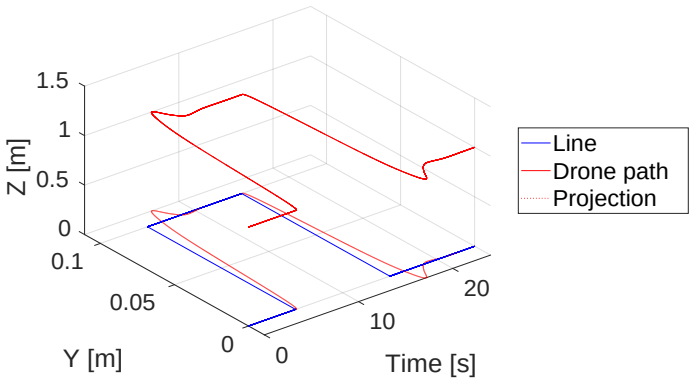


Figure 5: **Matlab simulation results:** Tracking of an abrupt changing line with gains $K_p = -130$ and $K_d = -100$. The control is only performed in the y-direction and the fictional z component is only for visualization purposes.

4 LINE DETECTION ALGORITHM

This section details the vision-based line detection algorithm essential for the proposed control strategy, which relies

http://www.imavs.org/

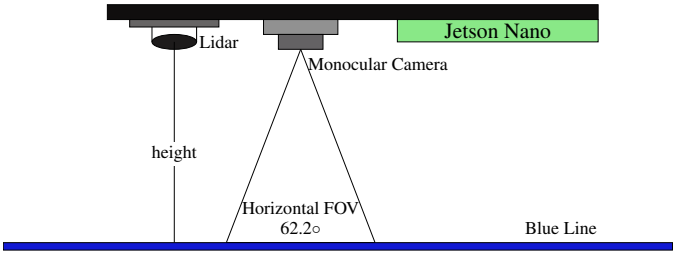


Figure 6: **Camera setup:** Positioning of the monocular camera and Lidar on the frame for effective data collection.

Table 1: Sony IMX219 camera parameters.

sensor width w_s	**sensor height** h_s	**camera focal length** f
3.68 mm	2.76 mm	3.04 mm

on continuous distance measurements from the camera to the line.

4.1 Camera Setup

To achieve a wide field of vision for detection, a camera sensor with a fixed focal length and wide lens angle is required. For example, a downwards pointing monocular Raspberry Pi Camera Module v2 with a Sony IMX219 CMOS sensor is considered in this study. The dimension of the camera are given in Table 1.

In a static image capturing mode, the camera has an active pixel area of 3280×2464 p, which can be used to determine the camera's horizontal and vertical field of view:

$$\begin{aligned} HFOV &= 2 \cdot \arctan\left(\frac{w_s}{2f}\right) = 62.37^\circ \\ VFOV &= 2 \cdot \arctan\left(\frac{h_s}{2f}\right) = 48.83^\circ \ . \end{aligned} \tag{17}$$

4.2 Contour Analysis and HSV Color Segmentation

The line-detection algorithm aims to extract the 2D coordinates of the middle of the line by identifying it's contour within the image frame. These coordinates enable the calculation of the distance d to the center of the the image.

The image obtained from the monocular camera is represented in the RGB color model, where each color is defined by its red, green, and blue spectral components. Luminosity significantly influences the accuracy and reliability of object detection within this model. To isolate therefore a specific color and improve the robustness of the detection algorithm, the image is converted to the HSV (Hue, Saturation, Value) color space. A mask is applied to this HSV image and filters out all colors except those within the specified color range. Additionally, morphological operations like erosion and dilation are applied on the mask to improve the image quality through noise reduction. These computer-vision algorithms increase the visual distinctness of the line, making it better discernible against the background.

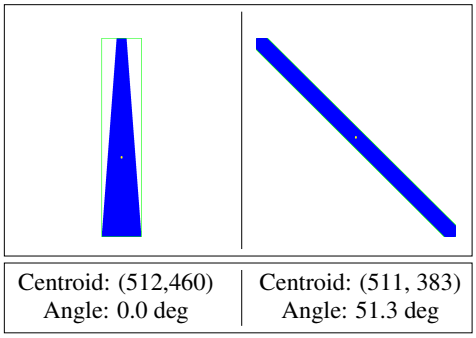


Figure 7: **Line detection:** Output of the line-detection algorithm for a straight line and a line with an angle w.r.t. the camera.

OpenCV functions are used to identify and filter contours from the processed mask, applying a specified minimum area threshold to exclude noise and irrelevant contours. Following this, the contour with the largest area is determined and approximated by a minimum area bounding rectangle. Parameters of this rectangle, like it's corner coordinates, allow to compute the centroid and the angle of the line. Figure 7 demonstrates a simulated scenario with the multicopter positioned at two distinct locations relative to the detected line.

5 HARDWARE AND SOFTWARE ARCHITECTURE

A big gap still exists between the proposed control strategy and a practical implementation. This section proposes a hardware and software architecture that allows to implement a autonomous line-following multicopter.

5.1 Hardware

The proposed multicopter hardware features a Pixhawk 6X Autopilot Flight Controller integrated into a multicopter frame similar to the Holybro X500 V2 quadcopter frame, which is the one used in our lab. A platform is added to the bottom of the multicopter's frame to allow placement of the camera and other sensors.

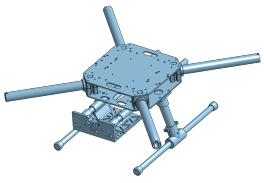


Figure 8: **Frame:** Holybro X500 V2 quadcopter

The Pixhawk autopilot system manages primary decision-making tasks, control, and guidance, while a companion computer (ex. a Jetson Nano) provides computer-vision capabilities such as object detection and line tracking. Figure 9 shows how the Pixhawk controller is linked to a telemetry module for data transmission to a ground station. An RC receiver

http://www.imavs.org/

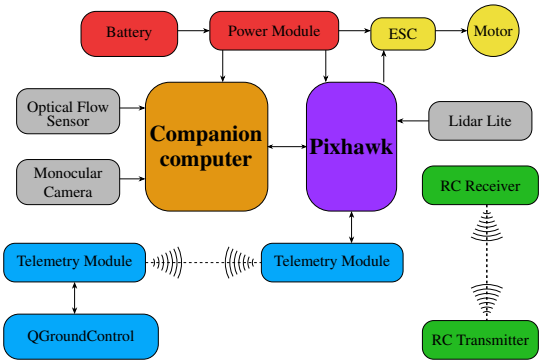


Figure 9: **Hardware:** Key components needed of the avionics architecture.

must be considered to allow switching to manual control for safety. Finally, a high accuracy range sensor like a Lidar-Lite can be used to achieve millimeter-level precise height measurements.

5.2 *Software Network*

The PX4 software contains a flight stack consisting of cascaded controllers with an estimation library that uses an Extended Kalman Filter for state estimation. Flight commands and sensor information are received by the flight controller from input interfaces such as the Radio control receiver, telemetry link, and the companion computer. Communication robustness between various components in the modular system is ensured by the MAVLink messaging protocol. The companion computer transmits the position and angle information to the Pixhawk flight stack through these MAVLink messages, as visualized in Figure 10.

The Pixhawk flight control architecture for the quadcopter consists of a cascade of P and PID controllers. The angle control and altitude control functionalities are managed by two separate Proportional(P) controllers at different locations in the pipeline, each operating at distinct frequencies of 250Hz and 50Hz respectively. Based on the received information, the Flight Control algorithm computes the attitude setpoint quaternion q_{sp} and sends it to the Pixhawk cascade controllers through the uORB subscribe/publish messaging protocol. By directly transmitting an attitude setpoint, the control system bypasses the initial stages of the Pixhawk cascade controllers, specifically the position (P) controller and the velocity (PID) controller.

6 PRACTICAL IMPLEMENTATION

In this last chapter, the proposed hardware and software is implemented in a high-fidelity simulator with realistic physics in order to close the gap between the simulation results and a real-world application. Results are obtained to demonstrate that the proposed vision-based control method performs successfully when integrated into a complete system.

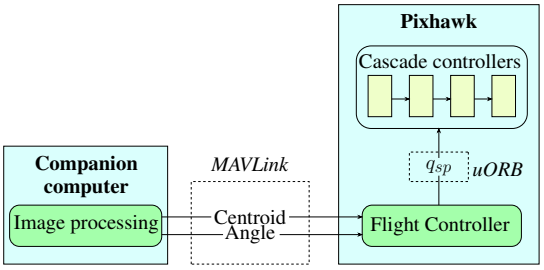


Figure 10: **Software:** Communication pipeline between the companion computer and the flight controller.

6.1 *Simulation pipeline*

The simulation made use of the Software-In-The-Loop (SITL) framework provided by PixHawk, which integrates the PixHawk autopilot flight stack with the Gazebo-Classic simulator. A simulated indoor environment featuring blue 19mm lines on the ground was recreated within Gazebo-classic, as shown in Figure 11. The 3DR Iris quadcopter model, already implemented in the PX4 Firmware, was used as the base multicopter model. A downward facing camera with the same specifications as the previously mentioned RaspberryPi Camera Module V2 and a range sensor were incorporated to this quadcopter model.

The video stream captured by the camera within the simulation is streamed via *gstreamer* to UDP port 5600. In parallel, a Python function established a connection to this port to retrieve the video, acting therefore as the companion computer. The OpenCV image processing library was used to compute the centroid and angle of the recorded line in each frame. The x and y coordinates of the centroid were then used to calculate the distance between the quadcopter and the line. This calculated data was subsequently transmitted back to the Pixhawk autopilot via a custom MAVLink message using pymavlink. Upon receipt, the Pixhawk autopilot translated the MAVLink message into a custom uORB topic for internal utilization. This streamlined communication pipeline enabled seamless communication between the Gazebo simulator, the image processing algorithm and the PX4 autopilot.

6.2 *Control algorithms*

The control algorithms were implemented in a custom Flight Task, which has been added to the Altitude Flight mode. Since the quadcopter is controlled through attitude setpoints, four separate control algorithms were used to control the thrust, pitch, yaw and roll inputs respectively.

The thrust controller was particularly crucial, as it was noticed that its performance influenced the roll controller's performance. The camera's field of view and image processing capabilities necessitate the line features to be stable and sufficiently large for accurate detection. Furthermore, without height compensation, roll inputs can cause the quadcopter to descend unexpectedly. A PID-controller, operating at a frequency of 100 Hz, was chosen and manually fine-tuned to

http://www.imavs.org/

achieve stable hovering at a desired height setpoint. Although good performance was achieved, its control performance was limited and was not without imperfections.

The roll controller, operating at 10Hz, was implemented as prescribed in Section 3. The measured angle of the line could be directly fed-forward to the yaw input, while the pitch input was simply held constant.

The resulting pitch, yaw and roll Euler angles were converted to a quaternion and together with the thrust setpoint were packed into the attitude setpoint. Finally, disabling PX4 position controller and transitioning to Altitude mode, enabling the custom Flight Task, allowed for the autonomous system to execute the developed control strategy.

6.3 Experiments and results

The performed simulation mimics the simulation of Section 3.3, where a straight line with an abrupt 1m change must be followed. A constant pitch value of $\theta = 0.008\ rad$ and a roll saturation of $\varphi = 0.3\ rad$ were used. These small angles were chosen to prevent the multicopter from exhibiting aggressive dynamics.

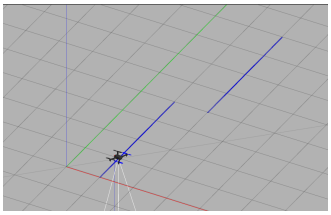


Figure 11: **Simulator:** View of the line configuration in the Gazebo simulator.

The obtained results from the simulation are illustrated in Figure 12. The multicopter is able maintain accurate tracking of the line and quickly correct the sudden 1-meter shift. Compared to the theoretical simulation in Section 3.3, where the coupling of roll control with other complex dynamics of the multicopter was not addressed, there is noticeably more oscillatory behavior observed. Overall, the drone is able to successfully track the proposed configuration, validating the proposed roll-control method. At the end of the simulation, when the multicopter is commanded to land and the controller is deactivated, the trajectory deviates from the blue markers.

By achieving consistent tracking in a realistic simulation of the fully integrated system, the control algorithm's effectiveness in autonomously guiding the multicopter using only visual input and range sensing is substantiated. Naturally, this assumes the availability of suitable line markers for the image processing algorithm.

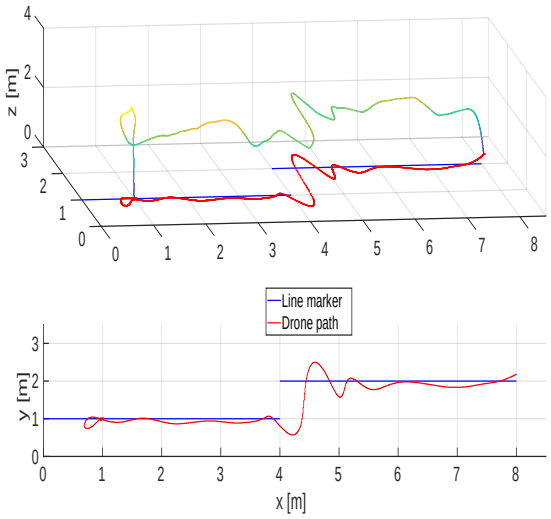


Figure 12: **Simulation results:** The colored path illustrates the three-dimensional trajectory of the multicopter as it follows the blue line markers on the ground, with the color gradient representing the drone's altitude. The red curves denote the projection of this trajectory onto the XY-plane.

7 CONCLUSION

This paper addressed the challenge of designing visual line-following autonomous guidance laws using roll setpoints instead of the commonplace yaw angle alternative. We show that such a system could not be closed-loop stable with classical PID controllers. However, we propose a feedforward term that renders the system closed-loop stable for a nonempty set of PID controller gains. In particular, we show and discuss a PD controller design. We first evaluate the efficiency of the proposed control strategy using a theoretical simulation in Matlab that includes the nonlinear dynamics of the multicopter and incorporates the Flight Management Unit (Pixhawk) controllers' architecture. Finally, the theoretical analyses are validated by practically implementing a quadcopter demonstrator in a high-fidelity simulation. The results show that the proposed control method is successful for line-following missions.

Having detailed the complete system design—from hardware and software to practical implementation—our future work will focus on transitioning this design to the x500 quadcopter present in our lab. The control method will be incorporated as the baseline autonomous controller with a state machine to perform other, more complex autonomous tasks. We aim to assess how the control method performs under real-world conditions, addressing any discrepancies between simulation and practice.

http://www.imavs.org/

REFERENCES

[1] https://2024.imavs.org/.

[2] Syed Agha Hassnain Mohsan, Muhammad Asghar Khan, Fazal Noor, Insaf Ullah, and Mohammed H Alsharif. Towards the unmanned aerial vehicles (uavs): A comprehensive review. *Drones*, 6(6), 2022.

[3] Hazim Shakhatreh, Ahmad H. Sawalmeh, Ala Al-Fuqaha, Zuochao Dou, Eyad Almaita, Issa Khalil, Noor Shamsiah Othman, Abdallah Khreishah, and Mohsen Guizani. Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges. *IEEE Access*, 7:48572–48634, 2019.

[4] Yago MR da Silva, Fabio AA Andrade, Lucas Sousa, Gabriel GR de Castro, João T Dias, Guido Berger, José Lima, and Milena F Pinto. Computer vision based path following for autonomous unmanned aerial systems in unburied pipeline onshore inspection. *Drones*, 6(12):410, 2022.

[5] Alexander Cerón, Iván Mondragón, and Flavio Prieto. Onboard visual-based navigation system for power line following with uav. *International Journal of Advanced Robotic Systems*, 15(2):1729881418763452, 2018.

[6] Maik Basso and Edison Pignaton de Freitas. A uav guidance system using crop row detection and line follower algorithms. *Journal of Intelligent & Robotic Systems*, 97(3):605–621, 2020.

[7] Amer Al-Radaideh and Liang Sun. Self-localization of tethered drones without a cable force sensor in gps-denied environments. *Drones*, 5(4):135, 2021.

[8] Hamid Didari Khamseh Motlagh, Faraz Lotfi, Hamid D. Taghirad, and Saeed Bakhshi Germi. Position estimation for drones based on visual slam and imu in gps-denied environment. In *2019 7th International Conference on Robotics and Mechatronics (ICRoM)*, 2019.

[9] AbdelRahman Eldosouky, Aidin Ferdowsi, and Walid Saad. Drones in distress: A game-theoretic countermeasure for protecting uavs against gps spoofing. *IEEE Internet of Things Journal*, 7(4):2840–2854, 2020.

[10] Dario Floreano and Robert J Wood. Science, technology and the future of small autonomous drones. *nature*, 521(7553):460–466, 2015.

[11] Tien-Loc Le and Nguyen Huu Hung. Trajectory tracking control of a line-following quadcopter using multilayer type-2 fuzzy petri nets controller. *Neural computing & applications*, 36:13617–13627, 2024.

[12] Shicheng Wu, Rui Li, Yingjing Shi, and Qisheng Liu. Vision-based target detection and tracking system for a quadcopter. *IEEE Access*, 9:62043–62054, 2021.

[13] Gomez A. Sotomayor J. and Castillo A. Visual control of an autonomous aerial vehicle for crop inspection. *Revista Politecnica*, 33(1), 2014.

[14] David García-Olvera, Armando Hernández-Godínez, Benjamín Nicolás-Trinidad, and Carlos Cuvas-Castillo. Line follower with a quadcopter. *PÄDI boletín científico de ciencias básicas e ingenierías del ICBI*, 7(14):30–36, 2020.

[15] Alexandre S. Brandão, Felipe N. Martins, and Higor B. Soneguetti. A vision-based line following strategy for an autonomous uav. 02:314–319, 2015.

[16] Alejandro Rodriguez-Ramos, Adrian Alvarez-Fernandez, Hriday Bavle, Pascual Campoy, and Jonathan P How. Vision-based multirotor following using synthetic learning techniques. *Sensors*, 19(21):4794, 2019.

[17] Torsten Merz, Simone Duranti, and Gianpaolo Conte. Autonomous landing of an unmanned helicopter based on vision and inertial sensing. In *Experimental Robotics IX: The 9th International Symposium on Experimental Robotics*, 2006.

[18] A. Masselli K. E. Wenzel and A. Zell. Automatic take off, tracking and landing of a miniature uav on a moving carrier vehicle. *Journal of Intelligent Robotic Systems*, 61(1-4):221–238, 2010.

[19] T. K. Venugopalan, Tawfiq Taher, and George Barbastathis. Autonomous landing of an unmanned aerial vehicle on an autonomous marine vehicle. In *2012 Oceans*, 2012.

http://www.imavs.org/

Swarm-in-Blocks: Simplifying Drone Swarm Programming with Block-Based Language

Agnes Bressan de Almeida
João Aires Correa Fernandes Marsicano
University of São Paulo
São Carlos, Brazil
joao.aires.marsicano@usp.br

http://www.imavs.org/

***Abstract*—Swarm in Blocks, originally developed for CopterHack 2022, is a high-level interface that simplifies drone swarm programming using a block-based language. Building on the Clover platform, this tool enables users to create functionalities like loops and conditional structures by assembling code blocks. In 2023, we introduced Swarm in Blocks 2.0, further refining the platform to address the complexities of swarm management in a user-friendly way.**

As drone swarm applications grow in areas like delivery, agriculture, and surveillance, the challenge of managing them, especially for beginners, has also increased. The Atena team developed this interface to make swarm handling accessible without requiring extensive knowledge of ROS or programming. The block-based approach not only simplifies swarm control but also expands educational opportunities in programming.

***Index Terms*—swarm robotics, block programming, drone swarms, high-level interface, multi-robot coordination, educational robotics**

I. INTRODUCTION

Swarm robotics is a field of multi-robot systems where a large number of relatively simple robots cooperate to achieve complex tasks. Inspired by natural systems such as ant colonies and bird flocks, swarm robotics emphasizes the importance of decentralized control and local interactions among robots [5]. The Swarm-in-Blocks project introduces a high-level interface based on block programming, designed to make drone swarm programming accessible and straightforward, even for users with limited technical knowledge [1].

II. PROJECT BACKGROUND

Swarm-in-Blocks originated as a project for CopterHack 2022, designed as a high-level interface using block language to simplify the programming of drone swarms. Each script in this language represents functionalities such as conditional structures, loops, or functions that handle swarm instructions. The platform is built on existing systems from COEX, specifically Clover. In 2023, Swarm-in-Blocks evolved into Swarm-in-Blocks 2.0, aiming to tackle more complex challenges associated with swarm robotics in an accessible and polished manner [2].

III. SYSTEM ARCHITECTURE

The Swarm-in-Blocks framework provides a secure and scalable platform for swarm robotics, consisting of the following components:

- **Block-Based Programming Interface:** A user-friendly, high-level programming interface that allows users to create scripts by fitting together code blocks like puzzle pieces.
- **Robotic Nodes:** Drones equipped with sensors and communication modules to interact with the programming interface and each other.
- **Control System:** Manages the execution of scripts and coordination of drone movements based on the programmed blocks.
- **User Interface:** Tools for monitoring and managing the swarm.

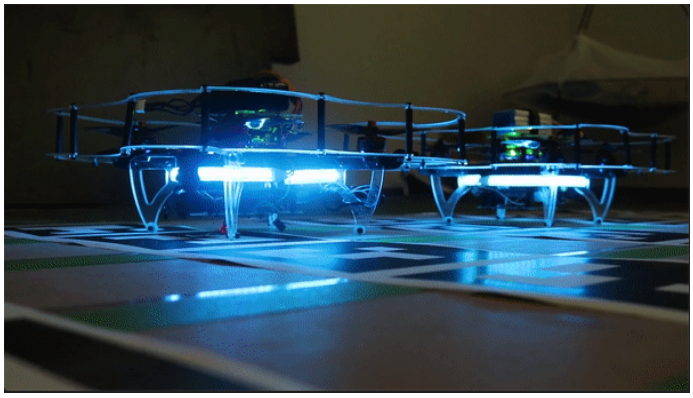


Fig. 1. Simple demo of the platform running.

A. Block-Based Programming Interface

The block-based programming interface is designed to be intuitive, enabling users to create complex swarm behaviors without needing to write traditional code. The interface includes blocks for various functions such as movement commands, conditional logic, and sensor data processing. This approach lowers the barrier to entry for programming drone swarms, making it accessible to a wider audience [3].

B. Robotic Nodes

Each drone in the Swarm-in-Blocks system acts as a robotic node, equipped with sensors for navigation and communication modules to interact with other drones and the control system. The drones are designed to be robust and capable

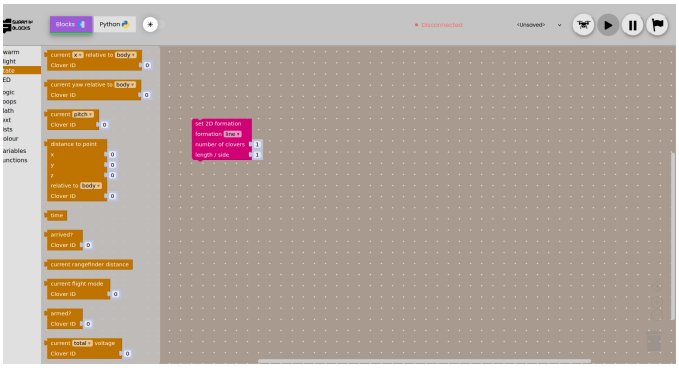


Fig. 2. Swarm Clover Blocks example.

f operating in diverse environments, from indoor spaces to utdoor fields [4].

. Control System

The control system is responsible for executing the scripts reated using the block-based programming interface. It interrets the block sequences and translates them into commands hat are sent to the drones. The system also handles coordiation between drones, ensuring that they work together to chieve the desired objectives [5].

). User Interface

The user interface provides tools for monitoring and manag-ng the swarm. Users can view the status of individual drones, ack their movements in real-time, and adjust parameters as eeded. The interface is designed to be intuitive and user-riendly, making it easy for users to manage complex swarm perations [3].

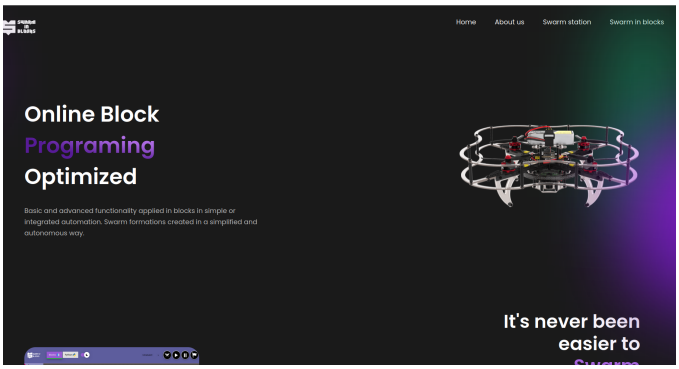


Fig. 3. Swarm in Blocks homepage

IV. APPLICATIONS

Swarm-in-Blocks has several potential applications, includ-ng:

. Industrial Automation

Swarm robotics can optimize industrial tasks like inventory anagement and quality control. The Swarm-in-Blocks frame-ork enables autonomous drones to navigate warehouses, perform stocktaking, and detect damaged goods, reducing manual labor and enhancing efficiency [4].

B. Agriculture

In agriculture, drone swarms can handle precision tasks such as crop monitoring, pest control, and irrigation management. The block-based interface simplifies the setup of these operations, improving resource use and crop yields through targeted interventions [5].

C. Entertainment

Swarm-in-Blocks facilitates the creation of aerial light shows by simplifying the programming of drone choreographies. The intuitive interface allows artists to design and execute captivating performances with ease [3].

V. BLOCK PROGRAMMING FOR GENERAL ROBOTICS

The ”Swarm-in-Blocks” project, developed for the Copter-Hack competition and enhanced in 2023 with Swarm-in-Blocks 2.0, offers a block-based programming interface for controlling drone swarms. This project not only advances swarm control technology but also holds significant educational potential, particularly as a teaching tool.

Block-based programming, widely used in educational robotics, simplifies the learning of programming. Platforms like Blockly and Scratch allow users to stack blocks representing code segments, making it easier for students of all ages to learn coding and computational thinking skills.

A. Educational Robotics

Educational robotics uses robots to facilitate learning in STEM education. Tools like Scratch and Blockly make programming accessible to young learners by allowing them to drag and drop blocks representing commands and logic structures [8].

B. Simplifying the Learning of Programming Concepts

Block programming makes learning programming concepts intuitive and accessible. It is particularly effective for introducing students to programming without the complexity of traditional coding languages.

C. Promoting Critical Thinking and Problem-Solving

Programming drones in formation fosters critical thinking and problem-solving skills. Students plan, test, and adjust instructions, considering variables like speed and altitude, enhancing their logical and systematic thinking.

D. Multidisciplinary Integration

Swarm-in-Blocks integrates concepts from mathematics, physics, computer science, and engineering. This multidisciplinary approach helps students apply theoretical knowledge to practical situations, essential in STEM education.

http://www.imavs.org/

http://www.imavs.org/

. Preparation for the Job Market

By introducing drone programming and blockchain technology, Swarm-in-Blocks prepares students for future job markets. Practical experience in these areas offers a competitive dge in robotics and blockchain careers.

. Applications in Educational Settings

Robots like LEGO Mindstorms and VEX Robotics are used n educational settings to teach programming and robotics. hese robots work with block-based coding environments, implifying the learning process [10], [9].

. Impact on Learning

Research shows that block-based programming enhances tudents' understanding of programming, improves problem-olving skills, and increases engagement in learning [8], [9].

VI. SWARM IN BLOCKS SYSTEMS IN DETAILS

Due to the complexity of the Swarm in Blocks project, was divided into systems, which represent a macroscopic iew of sets of similar functionality. In this section, we will etail the developed systems, dealing not only with the features eveloped, but also with the motivation and implications for nis.

. Formations

The "Formations" of the Swarm-in-Blocks project details ne use of 2D and 3D formations for drone swarms. The ython script formation.py calculates drone positions for formations such as lines, circles, squares, triangles, cubes, pyramids, and spheres. Using the Numpy library, it generates arrays f coordinates for each drone. The swarm.py file handles the pplication of these formations, allowing users to select and isualize formations through a menu interface. This facilitates ne creation and management of complex drone formations vith ease.

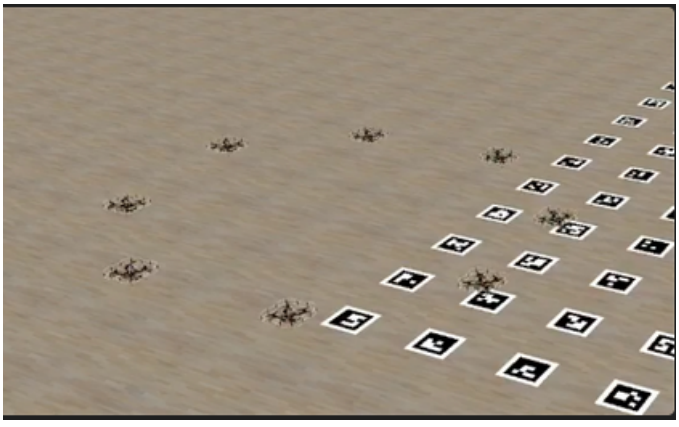


Fig. 4. Circle formation.

B. Transformations

The "Transformations" section of the Swarm-in-Blocks project covers operations that allow editing the drones' arrangement and the formation itself. Using matrix operations, procedures for translation, scaling, and rotation of the entire formation are developed. These transformations adjust the current coordinates of the drones, with special care for rotation, which requires centralizing the formation at the origin of the map. The collision prevention algorithm ensures that potential collisions are avoided during transformations.

C. Collision Avoidance

The "Collision Avoidance" section of the Swarm-in-Blocks project details an algorithm designed to prevent drone collisions within a swarm. The algorithm addresses three scenarios: one stationary and one moving drone, both drones moving on parallel paths, and both drones moving on non-parallel paths. It utilizes matrix operations and ROS (Robot Operating System) functionalities to manage drone positions and ensure safe distances are maintained, preventing collisions during swarm operations.

D. LED Effects

In the Swarm-in-Blocks project, the LED effects system allows for visual enhancements of drone formations. Ready-to-use effects include options for all drones, random drones, even and odd drones, and 2D formations. These effects can be controlled via user input, offering various modes like fill, fade, flash, blink, blink fast, wipe, rainbow, and rainbow fill. Each effect alters the LED behavior to enhance visibility and aesthetics, with some effects tailored to specific formations.

E. First Person View (FPV)

Part of the Swarm-in-Blocks 2.0, the First Person View (FPV) system enhances user experience by allowing real-time control of drones via a camera feed. Developed using roslibjs for web integration with ROS, it enables users to view and control drones through a web interface using JavaScript, HTML, and CSS. Users can select drones, view live footage, and control drone movements with keyboard inputs, improving flight safety and control.

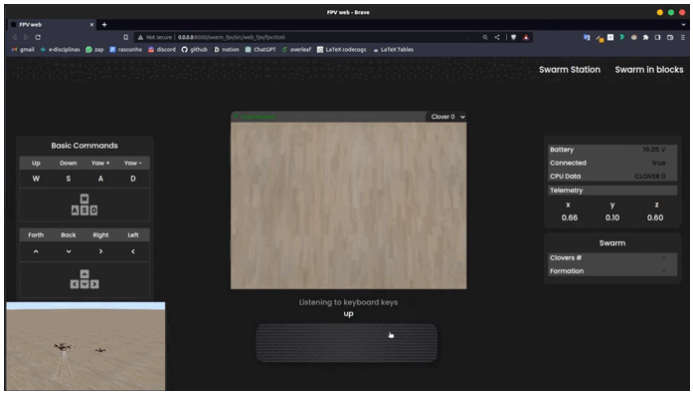


Fig. 5. Swarm in Blocks 2023 FPV.

. Smart Preview

Swarm Preview in the Swarm-in-Blocks project enables sers to visualize swarm behavior in simulations or real life ith minimal computational power. It's used primarily in lanning Mode but also in Simulation and Navigation Modes. sers can choose 2D or 3D previews via the Python or Blocks PI. Functions include resuming, saving, and canceling simuations, and navigating through operations with control arrows o identify issues.

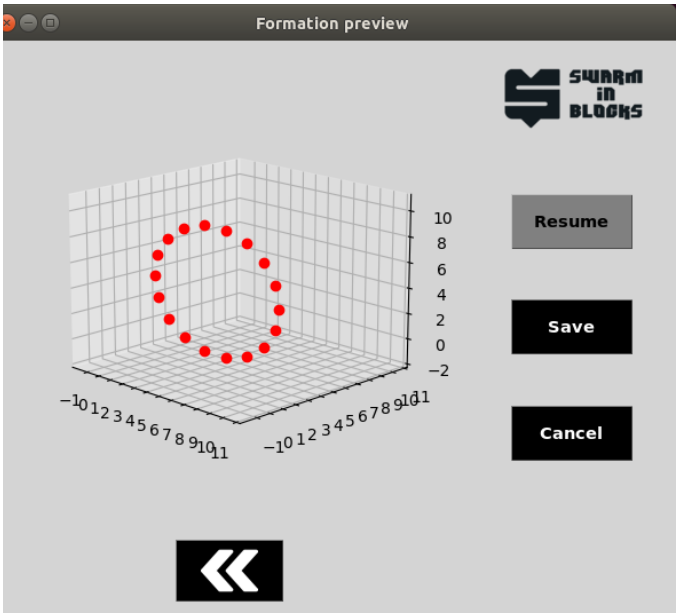


Fig. 6. 3D Plot Preview menu.

VII. CODE REVIEW

In this section, we delve deeper into the codebase that underins the core components of our Swarm-in-Blocks platform. he primary focus will be on examining the code responsible or the key pillars of our application: the **Swarm Clover locks**, the **homepage**, the **First Person View (FPV)** and he **Swarm Station**. These pillars form the foundation of our latform, each contributing unique functionality and ensuring eamless interaction between users and drone swarms.

The subsections that follow will present detailed explanaons and code snippets to illustrate how these pillars are onstructed, focusing on the logic and structure that enable he Swarm-in-Blocks platform to function effectively.

. Swarm Clover Blocks

The `swarm_clover_blocks` is a crucial component f the Swarm-in-Blocks platform, serving as the backend hat enables block-based programming for drone swarms. his backend application extends the functionalities of the riginal `clover_blocks` package, integrating support for ontrolling multiple drones simultaneously, which is essential or swarm operations.

1) Code Structure: The `swarm_clover_blocks` package is organized into several directories and files, each playing a specific role in the overall functionality of the platform:

a) `msg/`: This directory contains the message definitions used within the system. These messages are essentially the data structures that encapsulate user requests and responses, along with any necessary identifiers. The use of ROS messages allows for standardized communication between different nodes in the ROS ecosystem.

b) `programs/`: In this directory, pre-configured examples of block-based programs are stored. These examples are complete with the necessary blocks to perform specific tasks, providing users with ready-made templates that can be easily customized through the frontend interface.

c) `src/`: This is the core of the backend, where the primary logic of the `swarm_clover_blocks` resides. The key file here is `clover_blocks.py`, which orchestrates all backend operations, including the execution of user-defined programs and management of communication between the frontend and the drones.

d) `srv/`: The `srv/` directory defines the service files, which specify the structure of requests and responses for operations like loading, running, and storing programs. These services are essential for the dynamic interaction between the frontend interface and the backend logic.

e) `www/`: This directory forms the heart of the frontend. It includes the core Blockly files as well as custom adaptations specific to the Swarm-in-Blocks platform. Key files include `blocks.js`, which defines the available blocks and their functionalities, `index.html` for the web interface, and `main.css` for styling the interface. Additionally, `ros.js` and `roslib.js` are used for communication with the ROS backend.

2) Node Functionality: The `swarm_clover_blocks` node is the operational backbone of the block-based programming interface. It implements all the necessary services and topics to run Blockly-generated Python scripts. The node's functionality includes:

- **Service** `run` – Executes the Blockly-generated program, interpreting the block sequence as a Python script and commanding the drones accordingly.
- **Service** `stop` – Provides the ability to terminate an ongoing program, ensuring that drone operations can be halted safely and immediately.
- **Service** `store` – Allows users to save their block-based programs for future use, facilitating easy access and reusability of code.
- **Service** `load` – Retrieves stored programs from the `programs/` directory, making them available for execution through the frontend.

3) Parameters and Topics: The `swarm_clover_blocks` node also manages several parameters and topics that govern its operation:

a) Parameters: The parameters control various aspects of the drone's behavior, such as navigation tolerances, yaw angle precision, and confirmation prompts before running a

http://www.imavs.org/

rogram. These parameters can be adjusted directly in the node onfiguration or via URL GET-parameters when accessing the /eb interface.

b) Published Topics: The node publishes several topics, ıcluding:

- **`running`:** Indicates whether a program is currently being executed.
- **`block`:** Provides updates on the current block being executed, which is particularly useful for monitoring program progress.
- **`error`:** Reports any errors or exceptions that occur during program execution, ensuring that users are informed of issues in real time.
- **`prompt`:** Manages user input requests during program execution, allowing for interactive control of the drone swarm.

This comprehensive structure ensures that the `warm_clover_blocks` package can effectively manage ıe complex task of programming and controlling drone warms, providing both a user-friendly frontend and a owerful backend.

For more detailed information about the Swarm Clover locks, please refer to the official repository available at warm-in-Blocks Repository.

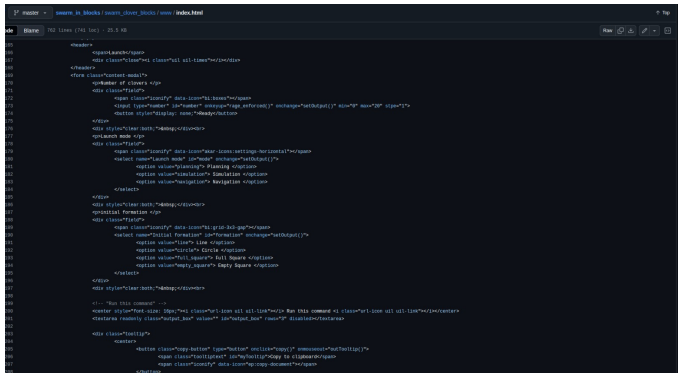


Fig. 7. Swarm Clover Blocks Code.

homepage

The `Clover_UI` serves as the main user interface for the warm-in-Blocks platform, providing users with a comprehenive homepage to launch and manage both individual drones nd drone swarms. This interface is accessible through a web rowser by navigating to the localhost address configured via ıpache, making it easy to control and monitor drones directly om any device on the same network.

1) Code Structure: The `Clover_UI` package is organized ıto a ReactJS project, with a clear structure that facilitates evelopment, deployment, and maintenance. The following are ıe key components of the project:

a) `cloverUI/`: This is the root directory of the ReacIS project. It contains all the essential files and folders needed o run the Clover UI, including the main HTML file and source ode.

b) `cloverUI/index.html`: This file is a crucial part of the ReactJS configuration. It acts as an entry point for the application, calling the main project components but not containing the actual application code.

c) `cloverUI/src/`: The `src/` directory is where the core logic and algorithms of the Clover UI reside. This is the primary source folder, containing all the React components and configuration files that drive the application's functionality.

d) `cloverUI/src/App.jsx`: The `App.jsx` file is the heart of the Clover UI project. It orchestrates the entire application by calling all the component pages. Each specific component's code is located in the `Components` folder, and `App.jsx` brings these components together to form the full user interface.

e) `cloverUI/src/Componentes/`: This folder contains the essential components that make up the application. Each component represents a specific part of the user interface, such as the header, carousel, or action buttons.

f) `cloverUI/src/style.js`: The `style.js` file contains Vite configuration settings that apply the defined styles dynamically. Vite is used to manage the project's build process, ensuring that styles are correctly applied during both development and production builds.

2) Frontend Technologies: The Clover UI leverages the ReactJS framework to create a responsive and interactive user interface. ReactJS was chosen due to its popularity, fast learning curve, and active community, making it an ideal choice for developing a robust and scalable frontend.

Additionally, the Clover UI takes advantage of complex visual effects, such as the Slider Carousel, to enhance the user experience. ReactJS's flexibility and ease of use enabled the development of a platform that is both functional and user-friendly, even though the page does not have its own backend.

To further streamline the development process, Vite, a ReactJS project builder, was used. Vite simplifies the creation and configuration of ReactJS projects, providing a pre-configured environment that is ready to use. This allowed the development team to focus on building the user interface rather than dealing with complex setup processes.

For more detailed information about the homepage, please refer to the official repository available at Swarm-in-Blocks Repository.

C. First Person View (FPV)

The development of the Swarm First Person View (sFPV) system allows drone pilots to control flights in real time through a camera installed on the drone. This feature has been restructured to run entirely on the web, integrating seamlessly with the Swarm Station. The sFPV system enhances drone operation by providing real-time visuals and data, such as battery levels, telemetry, CPU status, and flight state, thereby improving flight safety and control.

1) Code Structure: The `sFPV` package is organized into two main parts: the backend and the frontend. Each plays a critical role in ensuring the system's functionality:

http://www.imavs.org/

a) `msg/`: This directory contains the messages used to andle user requests and manage communication within the ystem. These messages are essential for relaying commands nd responses between the frontend interface and the backend ervices.

b) `src/`: The `src/` directory houses the core script les needed to run the FPV application. These scripts are esponsible for capturing the drone's camera feed, processing , and relaying it to the user interface for real-time display.

c) `launch/`: The `launch/` directory contains the nec-ssary launch files to execute the FPV system. These files llow users to initiate the application from the terminal, setting p the environment and connecting the necessary ROS nodes or the operation of sFPV.

2) Frontend: The frontend of the FPV application is built sing `roslib.js` to facilitate communication with the ROS ackend. The web interface, developed with HTML and CSS, llows users to control the drone through their browser. This nterface displays the live video feed from the drone's camera, long with essential telemetry data, enabling precise and safe rone operation. The user can interact with the drone using omputer keyboard inputs, adding another layer of control irectly from the web interface.

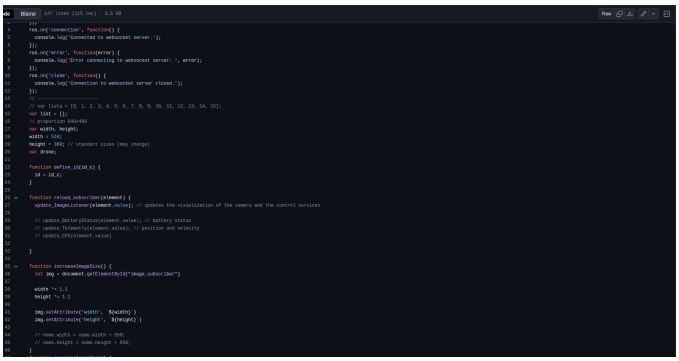


Fig. 8. First Person View Code.

For more detailed information about the sFPV, please refer o the official repository available at Swarm-in-Blocks Repos-tory.

). *Swarm Station*

The `swarm_station` package is responsible for run-ing the backend of the Swarm Station, which provides a ully integrated control center for managing drone swarms. his package utilizes the `tf2_web_republisher` package longside the `roslibjs` and `ros3djs` libraries to serve a D web visualizer. The system runs almost entirely on the eb, allowing users to access the Swarm Station from any evice connected to the same network as the host.

1) Code Structure: The `swarm_station` package is or-anized into several key directories, each playing a crucial role n the overall functionality of the Swarm Station:

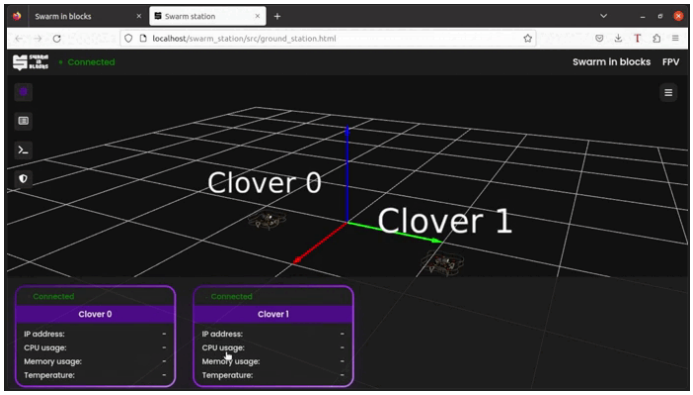


Fig. 9. Swarm in Blocks Swarm Station.

a) `launch/`: The `launch/` folder defines and initial-izes nodes, services, and parameters necessary to start the application. It also includes specific configurations required for the package.

b) `meshes/`: This directory contains definitions of the physical and dynamic characteristics of the Clover's 3D model used for simulation purposes.

c) `src/`: The `src/` directory is the core of the package, where source code is developed and compiled. It includes the definitions responsible for creating nodes, services, and other functionalities related to the frontend of the Swarm Station.

2) Main Features: The Swarm Station includes several integrated features designed to streamline swarm management:

a) Information Center: A central hub for displaying essential data on each drone, including an emergency "land all" button for quick response.

b) Drone's Process: Displays hardware information from each Clover's Raspberry Pi, making it easier to identify and compare potential issues.

c) Topic List: A feature that lists currently active topics, allowing users to monitor and analyze ongoing processes without needing an additional terminal.

d) Web Terminal: A web-based terminal that allows users to send commands directly from the web interface, avoiding the need for multiple screens. Multiple terminals can be opened simultaneously for handling various processes.

e) Safe Area: A feature that lets users define a safe operating area for the drones. If a drone exits this area, it is automatically landed, enhancing safety.

For more detailed information about the Swarm Station, please refer to the official repository available at Swarm-in-Blocks Repository.

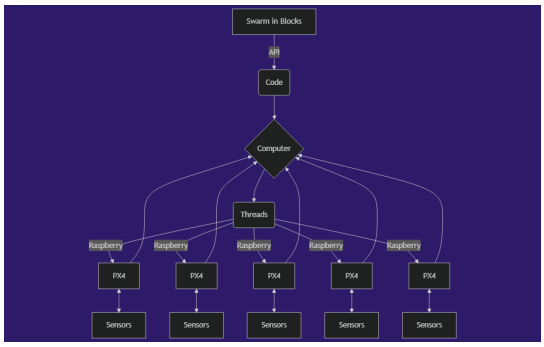
VIII. COMMUNICATION GRAFICS

The Swarm-in-Blocks platform effectively integrates MAVROS, ROS, and its web-based interface to manage and control multiple drones. MAVROS acts as a bridge between the drone's flight controllers and the ROS ecosystem, translating drone-specific commands and telemetry into a ROS-compatible format.

http://www.imavs.org/

ROS serves as the central hub, coordinating the flow of ıformation between the drones and the user interface. The warm-in-Blocks interface, built using `roslibjs`, commuicates with ROS nodes in real-time, allowing users to control rones through an intuitive block-based programming environent.

Commands created in the interface are converted into ROS ommands and sent to MAVROS, which executes them across ıe connected drones. The drones' telemetry and status updates re fed back through this communication chain, ensuring that sers can monitor and adjust operations as needed.



ig. 10. Schematic of communication flow between MAVROS, ROS, and warm-in-Blocks.

This streamlined communication framework ensures cohe-ive and synchronized control of the entire drone swarm.

IX. SIMULATION ANALYSIS

Simulation analysis is essential for validating and opti-ıizing robotic systems before real-world deployment. By sing advanced tools like Gazebo, we can recreate complex cenarios, evaluate system performance, and identify potential ssues in a virtual environment. This section focuses on ıtegrating two systems within Gazebo, covering the setup rocess, system interaction, and outcome analysis to refine esign and functionality.

This method provides a thorough evaluation of system ehavior under various conditions, allowing developers to nhance reliability and efficiency. Whether testing a single obot or a complex multi-agent scenario, the insights from ıese simulations are crucial for guiding development and nsuring successful deployment.

. *The Systems*

The first system features an Nvidia RTX 3060 mobile GPU, n Intel i7-11800H processor, and 16GB of DDR4 RAM, ffering a strong blend of graphical and computational power or smooth, high-resolution simulations in Gazebo. The second ystem, with an AMD Ryzen 5 3600 CPU, a Radeon RX 5700 T OC GPU, and 16GB of DDR4 RAM, provides robust rocessing and excellent graphical rendering, making it well-uited for intensive simulation tasks requiring both CPU and PU performance.

B. Real Time Factor

The Real-Time Factor (RTF) in Gazebo serves as an indicator of how efficiently the simulation runs compared to real-time. Ideally, an RTF of 1.0 signifies that the simulation is running exactly in real-time, while an RTF below 1.0 indicates that the simulation is running slower than real-time, typically due to increasing computational demands.

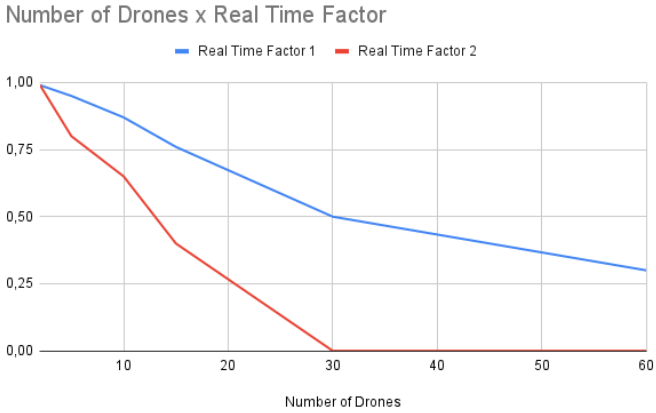


Fig. 11. Comparisson of RTF between 2 systems.

.

In the graph provided, the blue line represents the performance of the AMD-based system (AMD Ryzen 5 3600 with Radeon RX 5700 XT OC), while the red line represents the Nvidia-based system (Intel i7-11800H with Nvidia RTX 3060 mobile). As the number of drones increases, the RTF decreases for both systems, reflecting the higher computational load.

However, a critical observation is that when the number of drones exceeds 10, the RTF for the Nvidia-based system (red line) drops precipitously and eventually crashes the system, as indicated by the RTF reaching near zero. This crash highlights the Nvidia-based system's inability to handle the simulation workload as effectively as the AMD-based system, which, although also experiencing a decline in RTF, manages to sustain the simulation for a larger number of drones. This suggests that the AMD system provides better stability and performance under high-load conditions in Gazebo, whereas the Nvidia system struggles and ultimately fails when the number of drones increases beyond 10.

C. Analysis

The scalability of the system is directly influenced by the performance of its central node. The AMD-based system (Ryzen 5 3600 with Radeon RX 5700 XT OC) demonstrates better scalability, as it can manage an increasing number of drones without crashing, making it suitable for expanding the simulation's complexity. In contrast, the Nvidia-based system (i7-11800H with RTX 3060 mobile) shows limited scalability, struggling with more than 10 drones, which leads to system crashes and hinders the ability to scale the simulation effectively.

http://www.imavs.org/

To achieve greater scalability, upgrading to more powerful nd modern components would enhance the system's capacity o handle larger simulations and real life number of drones. 'his would ensure that the central node maintains performance nd stability as the number of drones and the complexity of the imulation increase, allowing for a more scalable and robust olution in demanding environments.

X. RESULTS AND DISCUSSION

The Swarm-in-Blocks framework has shown significant nprovements in drone swarm coordination and ease of use. :s block-based interface allows for quick prototyping and erative testing, enabling users with minimal programming xperience to manage drone swarm operations effectively [3].

. *Scalability and Flexibility*

Swarm-in-Blocks is highly scalable, supporting large num-ers of drones without added complexity. The modular blocks ffer flexibility in defining and modifying swarm behaviors, ssential for large-scale applications like environmental mon-oring and disaster response [4].

'. *Safety and Reliability*

Decentralized control mechanisms enhance the system's afety and reliability. Autonomous operation and redundancy nsure the swarm continues functioning even with drone alfunctions [5].

'. *Performance Evaluation*

Swarm-in-Blocks handles real-time operations with minimal atency. The block-based interface simplifies programming, al-owing users to focus on high-level strategies. Benchmark tests onfirm efficient performance across various environments [3].

). *User Feedback*

Users have praised the intuitive design and ease of use. Ed-cational institutions find it particularly valuable for teaching warm robotics, bridging the gap between theory and practice 4].

XI. FUTURE WORK

Swarm-in-Blocks is continuously evolving, with future work ocusing on:

. *Integration with Other Platforms*

Integrating with other robotic platforms for seamless in-eroperability, enabling comprehensive solutions for complex asks [3].

'. *Enhanced User Interface*

Improving the user interface with more customization, real-me feedback, and monitoring tools to simplify large swarm anagement [4].

'. *Field Testing and Validation*

Conducting extensive field testing in diverse environments o ensure robustness and identify areas for improvement [5].

XII. CONCLUSION

The Swarm-in-Blocks project simplifies the programming of drone swarms using a block-based language, making it accessible to users with limited technical knowledge. The Swarm-in-Blocks framework provides a user-friendly and scalable platform for communication and coordination, with potential applications in areas such as environmental monitoring, disaster response, industrial automation, agriculture, and entertainment. Future work will focus on further optimizing the framework and exploring additional applications in various domains.

ACKNOWLEDGMENTS TO THE ATENA TEAM

We would like to express our deep gratitude to the Atena team, part of the SEMEAR Group (Solutions in Mechatronic Engineering and Robotics Application) at the University of São Paulo (USP), São Carlos campus.

Comprising over 100 members, including undergraduate and graduate students, and supported by professors, the Atena team participated in competitions such as CopterHack. Team members include Agnes Bressan de Almeida, João Aires Correa Fernandes Marsicano, Felipe Andrade Garcia Tommaselli, Gabriel Ribeiro Rodrigues Dessotti, José Carlos Andrade do Nascimento, Lucas Sales Duarte, Matheus Della Rocca Martins, and Nathan Fernandes Vilas Boas. Their dedication and collaboration were essential for the project's success.

For more information, visit the SEMEAR website.

XIII. REFERENCES

REFERENCES

[1] Swarm-in-Blocks GitBook. (n.d.). Retrieved from https://swarm-in-blocks.gitbook.io/swarm-in-blocks
[2] Swarm-in-Blocks GitHub Repository. (n.d.). Retrieved from https://github.com/Grupo-SEMEAR-USP/swarm_in_blocks
[3] DroneBlocks. (n.d.). Retrieved from https://droneblocks.io
[4] Droneblog. (2023). The Making of A Drone Swarm: Everything You Need To Know. Retrieved from https://www.droneblog.com
[5] U.S. GAO. (2022). Science & Tech Spotlight: Drone Swarm Technologies. Retrieved from https://www.gao.gov
[6] Control.com. (2022). How to Train a Robot: Block-Based Programming. Retrieved from https://control.com
[7] Robots.net. (2023). What Is Block Coding. Retrieved from https://robots.net
[8] International Journal of STEM Education. (2022). The effects of educational robotics in STEM education. Retrieved from https://stemeducationjournal.springeropen.com
[9] Springer. (2021). Teaching Digital, Block-Based Coding of Robots to High School Students. Retrieved from https://link.springer.com
[10] iRobot Education. (2023). Coding Robots, Learning Library STEM Outreach. Retrieved from https://edu.irobot.com

http://www.imavs.org/

Comparative analysis of ESC and Flight Controller Communication Protocols and their impact on brushless motor response and drone control

Vitor Garcia Ribeiro*, André Carmona Hernandes†, Guilherme Barros Villela‡, and Marcelo Becker§
Universidade de São Paulo, São Carlos, Brazil, and Universidade Federal de São Carlos, São Carlos, Brazil

http://www.imavs.org/

ABSTRACT

This study explores the impact of communication protocols, specifically PWM and OneShot 125, between flight controllers and electronic speed controllers (ESCs) on brushless motor response and drone control. The experimental setup includes a test bench equipped with BLHeli-S ESCs and an STM32-based flight controller programmed with proportional control. Experiments quantified response time and control precision under different protocols. Data on bench angle and motor throttle were analysed using computational tools such as MATLAB. Notably, the results reveal no significant advantage in changing communication protocols. Instead, effectiveness and responsiveness are determined much more by the control loop frequency than by the rate of different protocols.

1 INTRODUCTION

In recent years, unmanned aerial vehicles (UAVs), commonly known as drones, have driven research and development of new technologies aimed at improving their efficiency and performance [1]. Among these technologies, the advancement of communication protocols is significant, promising to enhance the speed, reliability, and precision in the transmission of critical data [2].

However, to the best of our knowledge, the communication between two crucial components for the precise control of these aerial devices, the electronic speed controller (ESC) and flight controller, remains an underexplored topic in academia, often limited to user manuals and hobbyists. The current literature lacks analyses that evaluate the isolated impact of different communication protocols on the performance of drone control.

This paper aims to address this gap by conducting a comparative analysis of the response of an experimental setup subjected to two communication protocols: Standard PWM and OneShot125, while keeping all other parameters constant. To achieve this, a test bench configured to simulate the angular control of a drone under disturbances was used. This decision was made due to the fact it can perform tests with the same set-up for a long period of time, thereby enhancing the overall protocol effect comparison. Therefore, this study aims to provide insights of the communication protocol isolated from control loop rate.

The paper has been divided into 5 parts: the first being this introduction, which provides the context and motivation for the study. The second one is concerned with the theoretical background. It will define the basis of our work, where the fundamental concepts will be introduced. The methodology used is detailed in the third section, including the experimental procedures and tools used. The fourth part focuses on the results and analysis, it will compare the theoretical assessment developed with the results achieved. The last part is the conclusion, where we will summarize the findings, discuss their implications, and suggest potential directions for future research.

2 THEORETICAL BACKGROUND

In this section, we address our problem in two main areas: an overview of Electronic Speed Controllers (ESCs), their firmware, and the communication protocols they use to interact with flight controllers, and a mathematical derivation to support further analysis.

2.1 ESCs and Firmware

Electronic Speed Controllers (ESCs) are essential components in the operation of UAVs, particularly in multirotor configurations such as drones. They are responsible for regulating the speed and direction of brushless motors, which they achieve by receiving control signals from the flight controller and modulating the power supplied to the motors [3]. The ESC interprets these signals and adjusts the motor speed accordingly, ensuring stable flight dynamics and manoeuvrability.

Modern ESCs come with various firmware options, among which BLHeli-S stands out due to its advanced features and ease of customization [4, 5]. This firmware provides improved throttle response, reduced noise, and enhanced compatibility with different communication protocols [6]. These attributes make BLHeli-S particularly suitable for

*E-mail: v.ribeiro@usp.br
†E-mail: andre.hernandes@ufscar.br
‡E-mail: villela.guilherme@usp.br
§E-mail: becker@sc.usp.br

http://www.imavs.org/

research that requires consistent and reliable motor control, as in the present study.

2.2 Communication Protocols

The communication between the flight controller and the ESCs is facilitated by several protocols, each with distinct characteristics. For this study, we focus on two protocols: Pulse Width Modulation (PWM) and OneShot125.

These protocols were selected due to their simplicity and relevance in practical applications, providing a focused and cautious approach to this relatively uncharted topic.

Standard PWM is one of the most traditional methods for communicating between a flight controller and an ESC [7]. It operates by sending a series of pulse signals at 20 ms rate, where the width of each pulse corresponds to a specific throttle position or motor speed. Typically, standard PWM signals range from 1 ms (minimum throttle) to 2 ms (maximum throttle) [8]. The main advantage of PWM is its simplicity, making it compatible with almost all ESCs and flight controllers [4]. However, the relatively slow update rate can introduce latency, which might affect the responsiveness of the drone's control system [8, 4, 9].

OneShot125 is a more recent protocol than PWM, designed to overcome some of its limitations. It reduces the pulse width to a range of 125 µs to 250 µs, significantly increasing the signal update rate [9]. OneShot125 also reduces the latency introduced by the control loop, theoretically enhancing the overall performance of the drone. However, the increased complexity and the need for compatible hardware can be considered as potential drawbacks.

By analysing these two protocols, this study aims to empirically evaluate these theoretical advantages and limitations.

2.3 Mathematical derivation

Given the aforementioned problem, we now derive a mathematical model to aid our analysis. This model will be derived using concentrated parameters, as a way to encapsulate dominant effects. It is focused on a test bench (Figure 1) previously constructed and validated in a prior academic work [10].

First, some hypotheses must be drawn:

1. The dominant poles of a brushless DC motor can be modelled as a simple brushed DC motor [11].
2. A bearing damping is significant and must be included in the analysis of the motor equation.
3. The damping term in the bench equation consolidates all velocity-related losses, including both bearing losses and aerodynamic drag on the rod.
4. Changes in thrust due to propeller movement are not considered significant. This is because their effects can be approximated by the damping term mentioned in hypothesis (3), thus not requiring separate consideration.
5. The thrust constant and torque/power constant of the propeller are assumed constant and independent of the Reynolds number. Although it appears that it may not hold, experimental tests on some propellers, in special the ones used (GWS 5"x3"), show that a constant term can be used [12].
6. For the tests in consideration, the air density is constant.
7. By using disturbance analysis, the term $\Delta\omega^2$ may be neglected. This assumption is based on the premise that the variations in angular velocity squared are minor and do not significantly impact the overall system dynamics.
8. The angular sensor has an antialiasing first-order low-pass filter with bandwidth at 13 Hz.

From these hypotheses, it can be derived that:

$$\dot{\omega} = \frac{K_\tau \cdot V_{dc}}{R \cdot J_m} \cdot u - \left(\frac{K_\tau \cdot K_\omega}{R \cdot J_m} + \frac{B_m}{J_m} \right) \cdot \omega - \frac{d}{J_m} \cdot \omega^2 \quad (1)$$

In which $\dot{\omega}$ is the rotor angular acceleration, ω is the rotor speed, K_τ and K_ω are the torque and speed constants of a dc motor, V_{dc} is the nominal supply voltage, u is the control action, in the $[0, 1]$ domain, J_m is the rotor plus propeller inertia, R is the dc motor resistance, B_m is the bearing damping and d is the consolidation of $\frac{1}{2} \cdot \rho \cdot C_Q \cdot A \cdot R_p^3$, which is the aerodynamic drag generated by the propeller.

It is worth mentioning that ρ is the air density, A is the propeller disk area, C_Q is the torque constant and R_p is the propeller radius.

By applying disturbance analysis on equation 1, we achieve:

$$\Delta\dot{\omega} = \frac{K_\tau \cdot V_{dc}}{R \cdot J_m} \cdot \Delta u - \left(\frac{K_\tau \cdot K_\omega}{R \cdot J_m} + \frac{B_m}{J_m} + \frac{2d\omega_H}{J_m} \right) \cdot \Delta\omega \quad (2)$$

where $\Delta\omega$ is the disturbed speed, Δu, disturbed control action, u_H the control action to achieve hover condition and ω_H the rotor speed in hover condition.

Equation 2 is easily solved and has the form presented in equation 3.

$$\Delta\omega = \frac{K_\tau \cdot V_{dc} \cdot \Delta u}{R \cdot J_m \cdot p}(1 - e^{-pt}) \quad (3)$$

with $p = \left(\frac{K_\tau \cdot K_\omega}{R \cdot J_m} + \frac{B_m}{J_m} + \frac{2d\omega_H}{J_m} \right)$. However, this assumes an instantaneous response given a change in control action, so we can extend equation 3 by including a time delay due to the communication protocol:

$$\Delta\omega = \frac{K_\tau \cdot V_{dc} \cdot \Delta u}{R \cdot J_m \cdot p}(1 - e^{-p(t-a)}) \quad (4)$$

in which a represents the included delay.

We may also look on how the current should behave, by applying the result from equation 4 to the dc motor model, we achieve:

$$K'(1 - c_1 \cdot e^{-p(t-a)})\Delta u = \Delta i \tag{5}$$

in which K' and c_1 will be constants derived from equation 4. As one may observe, the current behaviour follows a typical first-order system.

If we extend our analysis to the whole bench, as described in equations 6 and 7 we are left with:

$$T = \frac{1}{2} \cdot \rho \cdot C_T \cdot A \cdot R_p^2 \cdot \omega^2 \tag{6}$$

$$\ddot{\theta} = \frac{\ell \cdot (T_f - T_r) - B \cdot \dot{\theta}}{J} \tag{7}$$

Equation 6 relates the angular velocity of the propellers to the thrust force T they generate, where C_T is the thrust coefficient.

As for equation 7, it describes the angular acceleration of the rod, influenced by the difference in thrust between the front and rear propellers, where ℓ is the distance from the centre of the motor, B is the overall drag, J is the overall inertia and θ is the angle position of the rod in the test bench.

With a little bit of effort on algebra arrangement, combining the equations 2, 6 and 7, adding the sensor effect on the plant, and doing the z-transform, one may see the overall plant discretized as described by equation 8.

$$\frac{\Delta\omega}{\Delta U}(z) = \frac{A_1 z^4 + A_2 z^3 + A_3 z^2 + A_4 z + A_5}{(z-1)(z - e^{\frac{-B_m T}{J_m}})(z - e^{-pT})(z - e^{-T/\tau})} \tag{8}$$

Where T is the sampling time given by the control loop rate and $1/\tau$ is the sensor bandwidth. By renaming the poles, for simplicity's sake, we now may add the delay effect, that follows equation 9:

$$\frac{\Delta\omega}{\Delta U}(z) = \frac{(A_1 z^4 + A_2 z^3 + A_3 z^2 + A_4 z + A_5)(z - zd)}{z^2(z-1)(z-p_1)(z-p_2)(z-p_3)} \tag{9}$$

It can be seen that the delay factor changes the plant order and adds a zero, given that the delay is bigger than one control loop. Also, it is possible to see that a comparison with the same control loop rate is paramount given that the sampling time changes the poles values, and thus, changes the behaviour of the bench.

3 METHODOLOGY

3.1 Experimental Setup

For the experiments, a proprietary test bench (Figure 1) was used.

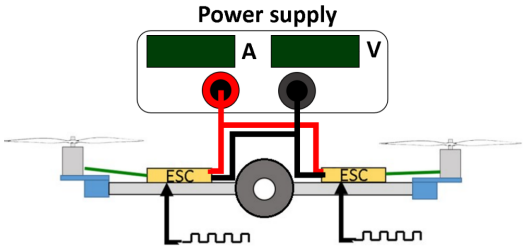


Figure 1: Test bench schematic [10]

The test bench includes a rigid aluminium rod balanced at its centre and connected to the device via a freely rotating axis supported by two bearings fixed to the base. At each end of the rod, brushless motors with attached propellers are mounted, and two BLHeli-S 30 A ESCs are fixed along the rod to control the motors.

At the centre of the structure is the power distribution system and a flight controller, based on the STM32F401 microcontroller. The STM connects via serial connection with a computer, to storage the data read.

A potentiometer serves as the primary sensor for angle measurement. The potentiometer is calibrated to provide a direct reading of the angular displacement, with the reference angle set at 0 degrees, which corresponds to the equilibrium position where the rod is parallel to the horizontal plane. Additionally, it has a low pass filter aimed at reducing noise from the power supply. This setup allows for precise assessment of the system's dynamic behaviour.

Lastly, the power supply is provided by a switched-mode power supply that provides continuous current to all components.

3.2 Motor current measurement

To attest any delay between the change in control action and the change in motor speed, it was initially inserted a 0.66 Ω - 50 W resistor in series with one of the motors. As a result, it was possible to measure the current flow on the motor.

As described by equation 5, a step disturbance on the controller is expected to result in the disturbed current following a first-order step response curve. Therefore, the proposed experiment involved activating both motors at the same speed (using 40% control action). After 6 seconds, to ensure steady-state conditions were achieved, the controller loop is initiated, using 50% control action as the hover condition.

When the loop starts, at the beginning of the control signal, a LED is toggled on each control step. This enables the measurement of the delay between the change in control action until the current first-order curve starts, marking when the motor received the command.

3.3 Controller Configuration

The controller used in this study employs a simple proportional control mechanism to maintain the desired angular

http://www.imavs.org/

http://www.imavs.org/

position of the test bench. It is effective enough for simple systems like this, eventually reaching a steady state without errors. A non PID approach was chosen to try highlighting the differences in control action under the different protocols, rather than using a more robust and generally faster PID controller. It is also worth mentioning that the same control rate (100 Hz) was chosen in all experiments.

The proportional controller operates with a gain value of $K_p = 60$, applied to the angular error measured in radians. Following that, the overall control action is divided in half. For the right motor, a hover value is added, and for the left, the hover value is subtracted from the split control action, thus, creating a correctional torque on the bench. The hover value used was 50% of the control action. This approach ensures that the control actions are symmetric and achieves the stability of the test bench.

Expressed in PWM units, the control signals have a resolution of 500 steps. This high resolution allows for fine adjustments to the motor speeds, with each step corresponding to a 0.2% change in the PWM signal. This resolution enables a precise control for maintaining the desired angular position and minimizing late oscillations.

3.4 *Experimental Procedures*

All experiments are conducted using a computer connected to the flight controller via a USB cable. A serial console application, such as PuTTY, is employed to communicate with the flight controller and to initiate one of the four pre-programmed[1] routines: two for calibrating the communication protocols and two for starting the stabilization procedure, which includes the proportional controller described in the section 3.3.

An experimental run begins with the initialization of the test bench, ensuring that all components are securely mounted and properly connected. Then, the rod is tilted to its maximum permissible angle (approximately 50 degrees from reference) to ensure the same starting point for all trials, given it has a physical barrier. Using the serial commands, we either choose calibration of PWM or OneShot when the change in protocol is needed, or simply run PWM or OneShot, depending on the run number.

The successful identification of the protocol by the ESC is confirmed through auditory signals (beeps) [6].

To ensure a smooth transition between initialization and the controller phase, an intermediate step is needed. Before the controller is activated, both motors are set to run at 40% power for 6 seconds to stabilize the rotor speed. Following this, a control stage consisting of 400 points is executed over 4 seconds at a frequency of 100 Hz.

To ensure data reliability and validity, 28 repetitions were done for each protocol, totalizing 56 runs. The order of experiments with the protocols were randomized. This was chosen because repetition reduces outlier impact and, by randomizing the protocols, it prevents drift to affecting one protocol or another [13].

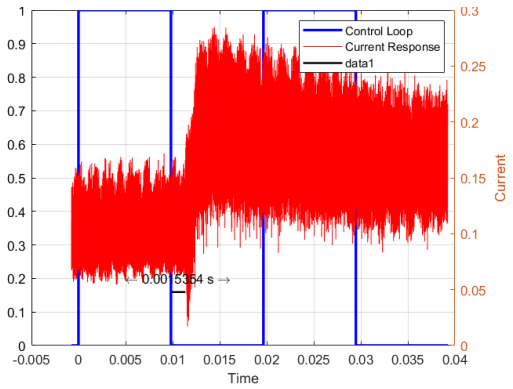
[1]The program code can be found in Appendix A:

4 RESULTS AND DISCUSSION

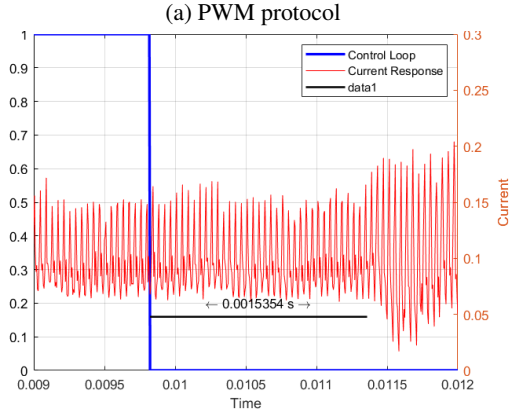
4.1 *Delay and Current Analysis*

After completing the experiments described earlier, the results underscore one difference between the protocols. Figure 2a illustrates the variation in current as a function of the control loop for the tests under the PWM protocol. For a more granular analysis, particularly to determine the response delay time, Figure 2b provides a detailed view. The same applies for the OneShot125 protocol, as depicted in Figure 3.

In these graphs, it is possible to roughly identify the first-order behaviour assumed previously.



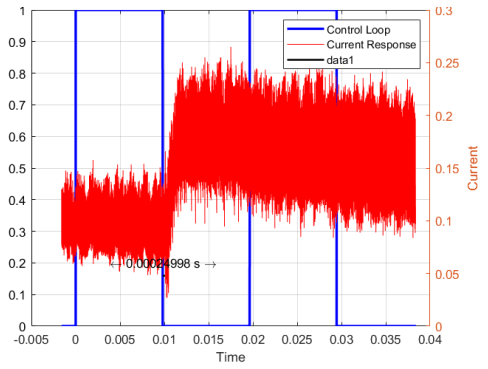
(a) PWM protocol



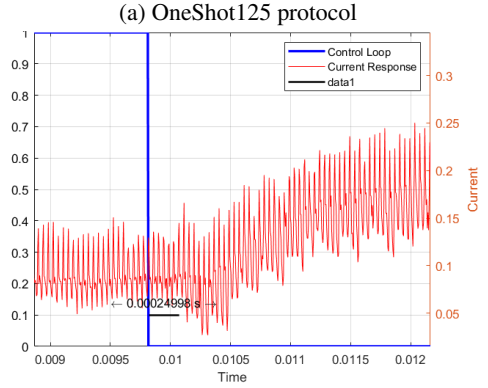
(b) PWM protocol in detail

Figure 2: PWM detected delay

Through careful examination, the delay in the PWM signal was approximately identified as being about 11.5 milliseconds after the first control loop, while for OneShot125, the delay is 10.25 milliseconds. Therefore, our experiments shown that different protocols have different delay responses.



(a) OneShot125 protocol



(b) OneShot125 protocol in detail

Figure 3: OneShot125 detected delay

Moreover, the delays perceived can be divided in two parts, one equal to the control loop rate, given that our signal implementation only updated the duty cycle on the next loop, and the other part, dependent on the protocol used. The delay phenomenon was modelled at equation 9.

4.2 *Comparative Analysis of Protocols*

In this section, we present the findings from the experiments conducted using the test bench setup. Figure 4 shows the consolidation of all 56 runs performed with both protocols. It starts the moment the controller is activated. The solid lines demonstrate the calculated mean of experiments, and the coloured region the standard deviation range.

From a preliminary analysis of the data, several observations can be made. First, there are about 400 milliseconds before the motors can outrun the counter torque and start tilting the rod - the primary hypothesis is a mass unbalance between both arms in the rod. Subsequently, the rod achieves a dynamic behaviour typical of a second-order underdamped system, stabilizing after a few seconds (not contained in this graph). This can be easily traced given a pole at the integrator position and the mechanical pole given by $e^{-B_m T/J_m}$.

It also appears that there is a small difference between both protocols, however because of the variability between trials, a statistical analysis is required to state it or not.

The error accumulated over time, arising from factors

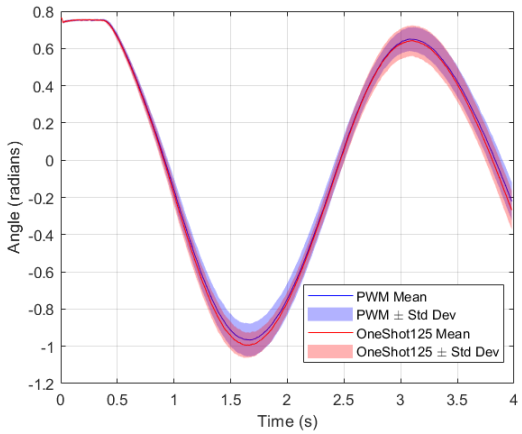


Figure 4: Mean angle of all runs with standard deviation ranges

such as control action rounding, angle measurements inaccuracies, dissipative aerodynamic effects, and other disturbances, poses a challenge for the comparative analysis between the two protocols. This is because a sudden surge in error at any given moment alters the system's response, which in turn affects subsequent readings and further modifies the control actions and overall system behaviour. This cumulative nature introduces a growing divergence between the system's expected and actual responses, which can mask the true effects of the communication protocols under study. As time progresses, these discrepancies become more pronounced, leading to an increase in the standard deviation between experimental runs, as observed in Figure 5.

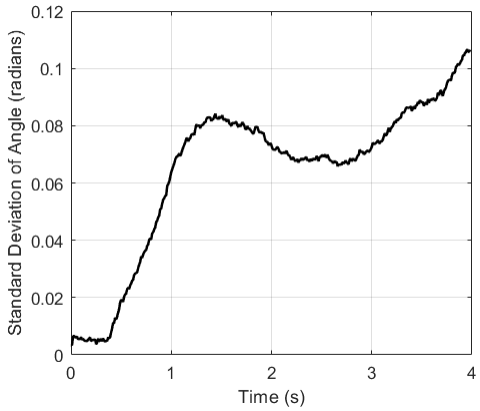


Figure 5: Standard deviation of the experimental data

Consequently, the longer the experiments run, the more difficult it becomes to attribute differences in system response solely to the communication protocol rather than to accumulating errors. This phenomenon suggests that the validity of the experiment, from a statistical perspective, is more reliable

http://www.imavs.org/

when focusing on the initial moments, where the accumulated error remains relatively small and controlled. This is one of the reasons it was decided to sample only 4 seconds of data. By limiting the analysis to this early phase, the intrinsic effects of the communication protocols are more accurately isolated, ensuring a more precise comparison of their performance without the confounding influence of escalating errors.

For these reasons, we focus our analysis on the rise time required to first achieve the horizontal plane and compare the two means using a Student's Two-Sample T-Test. This statistical test, given by equation 10, is particularly useful for determining whether there is a significant difference between the means of two independent samples [13, 14].

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}} \tag{10}$$

Where $\bar{X}_1$ and $\bar{X}_2$ are the observed means, S_1^2 and S_2^2 are the variances, and n_1 and n_2 are the sample size of each sample.

From the data acquired, the value of t was calculated as:

$$t = \frac{0.918 - 0.909}{\sqrt{\frac{0.0484^2}{28} + \frac{0.0515^2}{28}}} \approx 0.6738$$

This calculated t-value is far below the critical value for any reasonable confidence level, indicating that there is insufficient evidence to reject the null hypothesis [14]. In other words, the experimental results are not statistically significant to suggest any meaningful difference between the two protocols.

This conclusion holds for all points of analysis. As time progresses, the standard deviation increases, but the mean value remains close for both protocols, causing the t-value to decrease even further.

4.3 Interpretation of Results

The findings from our experiments reveal several key insights into the impact of communication protocols (PWM and OneShot125) on the system's response time and stability in a controlled test environment. Despite the initial hypothesis that OneShot125, with its faster communication rate, would result in a noticeably quicker response and more stable control compared to the standard PWM protocol, the experimental data do not support a statistically significant difference between the two, when all the other factors were held constant, and only the protocol was changed.

The delay measurements for both protocols suggest that while the OneShot125 protocol does indeed reduce communication delay (approximately 250 microseconds compared to 1.5 milliseconds for PWM), this difference does not translate into a significantly different overall system performance in the control plant developed for the present work.

Overall, as observed by equation 9, the delay part introduced by solely the protocol behaviour, apart from the 10 ms given next cycle update (which introduces a pole in zero), creates a pole in zero and a zero on the left of that pole, which has quick decay over time. Given that, it can be seen that our platform presents two dominant poles, one at $z = 1$, and integrator, and the other at $z = e^{\frac{-B_m T}{J_m}}$, which represents the overall mechanical pole from the rod, and, as one may see by pure inspection, the controlled (damped) frequency has a value of more or less 0.6 Hz ($\approx 1.5s$), and it is way more dominant than the delay effect.

Therefore, for slower systems where these dominant poles are near the integrator pole, the impact of communication delays is negligible. However, in cases where the mechanical poles are closer to the 0 pole, such as in high-performance applications like racing drones, these delays could become more consequential. Again, analysing Equation 9 after discretization, it becomes evident that the sampling period significantly impacts the system by altering all poles. Consequently, switching to a faster protocol, such as OneShot125, may be justified, as it allows for a control loop frequency of up to 1 kHz, compared to PWM, which is limited to a maximum of 250 Hz.

On a more practical note, the PWM can have a more refined range for the control action vector, since it needs 1 ms for the whole range, thus, a 0.1 microsecond timer resolution, will lead to a 0.01% step. On the other hand, to achieve the same precision with OneShot125, it will be needed a timer resolution of 12.5 nanoseconds, which corresponds to 80 MHz, thus, a minor change in frequency may induce a control action error.

4.4 Implications and Future Directions

The overall results suggest that the choice of the communication protocol should consider two aspects, the first one is the control loop rate, which should be guided by the dominant poles of the drone, and the other aspect should be control action resolution. Therefore, if the control engineer, given robustness, parametric variation, noise immunity decides that the drone may be controlled by a frequency less than 250 Hz (4 ms), the PWM may be chosen, given that the introduced delay does not interact with the dominant poles. This decision would favour the control resolution and a healthier hardware communication, and, as showed in our experiments, there is no difference compared to the OneShot protocol. On the other hand, if the control engineer decides it needs to be controlled with higher frequencies, one may decide to use the OneShot protocol.

Future research should consider extending the analysis to more complex operational conditions, including high-performance controllers, that are immune to disturbances, noise and parametric variations. Such controllers may need to affect higher frequencies and thus, the protocol delays may be a contributing factor. Additionally, investigating the impact of the communication protocol on other aspects of performance, such as energy efficiency or robustness to noise,

http://www.imavs.org/

could provide a more comprehensive understanding of their role in UAV systems.

Based on the findings of this study, it can be suggested that the assembly and component selection for an aerial vehicle should not overly prioritize the choice of communication protocol between ESCs, motors, and flight controllers. Likewise, the selection of hardware should not be limited solely to ensure compatibility with "modern" protocols. Instead, other factors should also be considered beyond the protocol's update rate. These include the availability of protocol-specific technologies that offer distinct advantages — such as telemetry feedback, regenerative braking, or active freewheeling, available in some protocols — as well as the capability (and necessity) to implement a more frequent control loop. Furthermore, considerations like the cost and efficiency of the hardware are even crucial and should only be impacted by the protocol's update rate if the application demands extremely response time and the system is very responsive - although it still needs to be confirmed by future research.

5 CONCLUSION

This study presents a comparative analysis of two communication protocols, PWM and OneShot125, and their impact on brushless motor response and drone control.

Different delays were detected between the protocols, with OneShot125 showing faster responsiveness. However, for the specific case of our experimental setup, the small communication delays have not shown significantly effect in the overall dynamics. Thus, the small improvements of OneShot125's reduced latency do not manifest in a substantial way under these test conditions. While the delays were detected and measured, their effect on most real-world performance is expected to be minimal. This conclusion is particularly valid for larger, slower drones, where such delays do not introduce noticeable differences. However, for smaller, faster, or more aggressive drones, these delays might have a more pronounced impact, requiring further exploration.

With the methodology hereby presented, we hope to enable future work to be carried out, with a focus on enhancing our understanding of existing protocols in different applications. Consequently, it will be possible to guide better decisions regarding practical UAV applications.

ACKNOWLEDGEMENTS

This work was supported by National Council for Development and Research (CNPQ), Brazil, grant no. 2023-1263. and by São Paulo Research Foundation (FAPESP) grant no. 2023/11661-0.

REFERENCES

[1] Syed Agha Hassnain Mohsan, Nawaf Qasem Hamood Othman, Yanlong Li, Mohammed H. Alsharif, and Muhammad Asghar Khan. Unmanned aerial vehicles (uavs): practical aspects, applications, open challenges, security issues, and future trends. *Intelligent Service Robotics*, January 2023.

[2] Drago Hercog. *Communication Protocols: Principles, Methods and Specifications*. Springer International Publishing, 2020.

[3] H. V. Rahul, R. Rachana, L. G. Vaishnavi, V. Nesara, G. K. Savyasachi, and Alfred Vivek D'Souza. Electronic speed controllers: A review. *International Journal of Engineering Research & Technology (IJERT)*, 10(11), 2022. ICEI – 2022.

[4] Oscar Liang. Overview of ESC Firmware and Protocols: How Flight Controllers and ESCs Communicate. https://oscarliang.com/esc-firmware-protocols/, nov 2023. Accessed: 2024-06-03.

[5] ArduPilot Dev Team. BLHeli32 and BLHeli-S. https://ardupilot.org/copter/docs/common-blheli32-passthru.html, 2024. Accessed: 2024-05-20.

[6] SiLabs. *Operation manual for BLHeli*. TME, Available at https://www.tme.eu/Document/565a6d126189bb1b31b17819c38dcc5c/Bullet%20firmware%20instruction.pdf, 2023. Accessed: 2024-05-12.

[7] Kristen N. Mogensen. Motor-control considerations for electronic speed control in drones. *Analog Applications Journal*, 4Q 2016:1–6, 2016.

[8] Jian Sun. *Pulse-Width Modulation*, pages 25–61. Springer, London, 2012.

[9] A.L. Imoize, S.M.N. Islam, T. Poongodi, L.K. Ramasamy, and B.V.V.S. Prasad. *Unmanned Aerial Vehicle Cellular Communications*. Unmanned System Technologies. Springer International Publishing, 2022.

[10] Vitor Campos and André Carmona Hernandes. Educational testbed for aerial angular control: Project and study case. In *LARS/SBR/WRE*, pages 522–527, 2018.

[11] Mark Cutler and Jonathan P How. Analysis and control of a variable-pitch quadrotor for agile flight. *Journal of Dynamic Systems, Measurement, and Control*, 137(10):101002, 2015.

[12] Robert W Deters. *Performance and slipstream characteristics of small-scale propellers at low Reynolds numbers*. University of Illinois at Urbana-Champaign, 2014.

[13] Nist/Sematech. *Handbook of Statistical Methods*. https://doi.org/10.18434/M32189, 2012.

[14] George Casella and Roger Berger. *Statistical inference*. CRC Press, 2024.

http://www.imavs.org/

APPENDIX A: RESOURCES

All relevant codes used in this project will be available in a GitHub repository at: https://github.com/VitorGaRi/BenchTestProtocolsComparison.

http://www.imavs.org/

Exploring the potential of collaborative UAV 3D mapping in Kenyan savanna for wildlife research

Vandita Shukla[1,2*], Luca Morelli[1,3 †], Pawel Trybala[1‡], Fabio Remondino[1§], Wentian Gan[4¶], Yifei Yu[4‖] and Xin Wang[4**]

[1]3D Optical Metrology Unit (3DOM), Bruno Kessler Foundation (FBK), Trento, Italy
[2] Computer Vision and Machine Learning Systems Group, Institute for Geoinformatics, University of Muenster, Germany
[3]Dept. of Civil, Environmental and Mechanical Engineering, University of Trento, Italy
[4] School of Geodesy and Geomatics, Wuhan University, People's Republic of China

http://www.imavs.org/

ABSTRACT

UAV-based biodiversity conservation applications have exhibited many data acquisition advantages for researchers. UAV platforms with embedded data processing hardware can support conservation challenges through 3D habitat mapping, surveillance and monitoring solutions. High-quality real-time scene reconstruction as well as real-time UAV localization can optimize the exploration vs exploitation balance of single or collaborative mission. In this work, we explore the potential of two collaborative frameworks - Visual Simultaneous Localization and Mapping (V-SLAM) and Structure-from-Motion (SfM) for 3D mapping purposes and compare results with standard offline approaches.

1 INTRODUCTION

Unmanned Aerial Vehicles (UAVs) have become an essential tool for supporting conservation challenges to collect data for 3D mapping, surveillance and monitoring [1, 2, 3, 4]. UAVs, however, have generally limited flying time due to battery power, requiring solutions that increase the efficiency of data collection in the limited airborne time. A possible solution is to use multiple UAVs (or agents/swarm) which can collaborate in the data collection to support studies on wildlife populations and habitat [5] and overcome the challenges induced by the use of a single platform through collaborative missions [6]. Collaborative mapping has shown its potential to study wide-ranging animal species that require larger area coverage or locating them in complex environments that may require a longer survey time [7][8]. Moreover, different agents in the mission can leverage different hardware and sensor strengths for fulfilling multiple goal targets [9][10] [11].

*vshukla@fbk.eu
†lmorelli@fbk.eu
‡ptrybala@fbk.eu
§remondino@fbk.eu
¶gwt2019@whu.edu.cn
‖yfyu2020@whu.edu.cn
**xwang@sgg.whu.edu.cn

Within the context of wildlife monitoring, a UAV-based collaborative approach has multiple benefits, including:

- extension of mission time of an on-going pursuit (e.g. real-time poacher or animal tracking);
- 3D geometric information recovery (e.g. 3D pose estimation from stereo vision from two synchronised UAVs);
- coverage of wider target area (e.g. for tracking multiple herds or speeding up mapping operations).

UAV surveys for wildlife studies have multifaceted goals [12], such as mapping the terrain while keeping a lookout for dynamic targets such as animal herds or poachers. Missions need frequent path re-planning due to changes in the objective priority, which can switch from mapping the scene to tracking targets. Therefore, upon locating an animal group of interest, diverting from original waypoint-based mapping path and navigating to unknown terrains is often required when the animals move. In such a scenario where one UAV has to divert to follow animals of interest, continuation of original survey can be achieved through collaborative mapping i.e. a follow-up from one or more UAVs. High quality real-time scene reconstruction and camera trajectory feedback can optimize the *exploration* vs *exploitation* balance of such a collaborative mission through enhanced situational awareness for each agent. Exploration and exploitation are the key phases of collaborative missions in which the information gained by one or more UAVs (*exploration*) is used for path planning optimization (*exploitation*) to achieve data acquisition with the least redundancy and best resource utilization [5].

Visual Simultaneous Localization and Mapping (V-SLAM) is a key technique in UAV-based 3D mapping. It allows an agent to localize itself within a 3D space in real-time while reconstructing the map (environment) through various sensors [13]. SLAM saw increase in popularity for visual sensor based mapping in 2004 with the showcase of MonoSLAM [14], a first complete real-time visual-SLAM system. A SLAM system consists of the following parts: front-end comprising sensor and odometry components; and back-end for loop optimization and mapping. Front-end acquires data from

http://www.imavs.org/

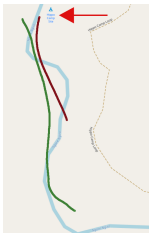
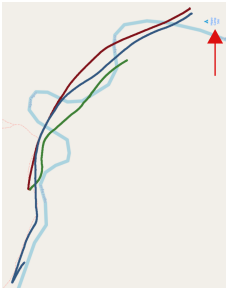
single or multiple agents[15] and, in case of V-SLAM, covers image processing (feature extraction, matching and tracking, and pose estimation on existing map), while back-end increases and optimizes the map. Multi-agent visual SLAM architectures are usually centralized: UAVs or unmanned ground vehicles (UGVs) acting as front-end agents focus on their real-time state estimation, and a server backend that performs aggregate map generation and loop closure. Collaborative V-SLAM's performance deteriorates in complex outdoor and dynamic environments, especially if RGB cameras are the unique odometry sensors. One of the main problems is that in the absence of loop closures the drift of the trajectory increases rapidly, while exploring larger areas with multiple agents. Systems equipped with other types of visual (RGB-D camera) and non-visual (IMU) sensors along with RGB cameras or LiDAR sensors are generally more reliable for mapping outdoor complex environments [16, 10, 9, 17].

Starting from image sequences, Structure-from-Motion (SfM) estimates three-dimensional structures, usually a sparse map of 3D tie points, and image poses [18]. When it comes to high-fidelity mapping with RGB data, SfM can be more accurate than V-SLAM, e.g. since SLAM forces a limit on the number of tie points and/or uses lower quality local features to improve computational time. However, traditional SfM is generally performed offline [19], in contrast to the real-time-oriented V-SLAM. Recent advancements propose real-time Structure-from-Motion, also called On-the-Fly (OtF) SfM [20]. OtF-SfM has demonstrated the capability to photogrammetrically process image streams acquired by multiple collaborative agents without the requirement of spatio-temporally ordered input, normally a major pre-requisite for onboard V-SLAM [21]. The performance of collaborative V-SLAM depends on individual UAV mapping performance, since the SLAM algorithm runs onboard each UAV independently, whereas optimization such as loop-closures and map-aggregation happen on a centralized back-end [16, 17]. On the other hand, collaborative SfM could reduce the dependence of collaborative mapping on individual agent mapping accuracy or on inter-agent communication for global optimization [16]. It can handle asynchronous image input from multiple UAVs and prior reconstructions are seamlessly merged on the server into a complete model whenever images can be registered, which motivates evaluation of its performance in challenging savanna environment.

In this paper, we aim to study the potential and limitations of UAV-based collaborative mapping in the context of wildlife conservation using only visual data. A real-time SfM approach (hereafter referred to as OtF-SfM [1]) and a collaborative V-SLAM software (hereafter referred to as CCM-SLAM [22]), have been compared to traditional offline SfM processing. In addition, the influence of learning-based tie points extracted with convolutional neural networks (CNNs)[23] trained for difficult scenarios is tested evaluating the final accuracy of flight trajectories.

2 DATA

The data used for the study was collected in July 2023 at the Ol Pejeta Conservancy, located in Laikipia County (Kenya), as part of the preliminary data collection mission of the WildDrone[2] EU project.

Dataset ID	Sample Image	Trajectories
1		
2		

red - flight 1 / **green** - flight 2 / **blue** - flight 3

Table 1: Used datasets with image samples and GNSS trajectories (L1 GPS positions stored by the onboard receiver) - different colors represent different agents. The red arrow points at the "Hippo Camp" on the map, a location later used to describe the flight directions.

As real collaborative datasets acquired in savanna are not yet available, for testing the collaborative UAV performance in these wild environments, we simulated multi-agent data input stream starting from some videos recorded in the same area. We used two datasets (Table 1), each representing one

[1] https://yifeiyu225.github.io/on-the-flySfMv2.github.io/

[2] https://wilddrone.eu

multi-UAV scenario, collected using a quadrotor DJI Mavic-3E. For dataset 1, image data comes from two flights representing two collaborative aerial agents, whereas in dataset 2 three flights represent three collaborative aerial agents. The flight data selected for each dataset (specifications in Table 2) fulfilled all of the following criteria:

- The flights were carried out sequentially or with no more than a 10-minute interval between the end of one flight and start of the next;
- the flight trajectories of individual flights were overlapping to test the performance accuracy in merging sparse map of independent flights;
- scenes with moving animals, cars, or people were not used, to focus tests on mapping accuracy .

Dataset ID	No of agents	V-SLAM frame rate	SfM frame rate	Image size (pixels)
1	2	29.9 fps	1 fps	1920 x 1080
2	3	29.9 fps	1 fps	1920 x 1080

Table 2: Dataset specifications. For SLAM evaluation we pre-created agent ROSBAG files with messages recorded at 29.97 fps which were played simultaneously on the client nodes to simulate real-time video streaming.

In Table 3 the number of frames used in each processing approach and the flight direction are reported.

Dataset	Agent	Frame Count (V-SLAM)	Frame Count (SfM)	Flight Direction
1	1	1580	55	Towards the Hippo Camp
	2	4685	162	Away from the Hippo Camp
2	1	6854	237	Away from the Hippo Camp
	2	6855	237	Towards the Hippo Camp
	3	6853	237	Away from the Hippo Camp

Table 3: Dataset 1 characteristics: simultaneous flights are simulated from two agents performing data acquisition from opposite viewpoints. Dataset 2 characteristics: three agents were simulated, all flying in the same region, one with different direction with respect to the other two.

This selection process enabled us to use the collected datasets to simulate a collaborative mission with multiple drones even if data was acquired by a single UAV performing sequential flights.

3 METHODOLOGY

To investigate the potential and limitations of collaborative mapping, the approaches are considered within two main domains: offline (3.1) and real-time (3.2) processing. Here offline processing refers to mapping performed after complete data acquisition i.e. methods that operate once all data (i.e. images) are available. On the other hand, real-time processing refers to methods that perform mapping while data are being acquired. For already acquired data, real-time processing can be evaluated by simulating a real-time data stream through ROS (for V-SLAM) or terminal clients (for OtF-SfM). To compare the performance of the tested mapping approaches, the available GNSS data (with an accuracy at meter-level in single-point positioning) was used to assess the accuracy of the agent trajectories in terms of Root Mean Square Error (RMSE), i.e. the error between the estimated and the observed GNSS positions.

3.1 Offline processing

We selected two well-established SfM software products: a commercial — Agisoft Metashape [3] — and an open-source — COLMAP [24], widely used for research purposes due to the significant level of control it offers in all stages of the photogrammetric pipeline. It is not fully known what type of local features for tie point extraction are used in Metashape wheareas COLMAP uses a GPU version of RootSIFT [25]. In SfM, distinctive and repeatable keypoints are identified in the images, descriptors are assigned to describe the neighbourhood of these points, and candidate corresponding points (tie points) are exhaustively matched across all pairs of images by comparing the similarity of the descriptors. The candidate matches are further refined using epipolar geometry, from which images can be oriented with different strategies. The incremental approach is considered in this work: after initialization with an image pair, subsequent images are oriented through a resection on the triangulated tie points and expansion of the 3D tie points via triangulation, followed by several local and/or global bundle adjustments.

Considering that reliable and accurate tie points are fundamental for a good orientation of an image block, keypoints extracted and matched with CNNs have also been tested in addition to the traditional local features implemented in Metashape and COLMAP. These CNN-based features were trained to overcome the limitations of SIFT-like methods, particularly in scenarios involving matching images taken at significantly different viewing angles and with drastic changes in lighting conditions. In the context of this study, the primary challenge is matching images captured by drones fol-

[3] https://www.agisoft.com/

http://www.imavs.org/

http://www.imavs.org/

lowing approximately the same trajectory, but in opposite directions, where the terrain is viewed at an angle that varies by approximately 180 degrees. Therefore, SuperPoint[26], an end-to-end detector and descriptor initially trained on synthetic images with a further refinement on real images, was also included in our tests. SuperPoint features are typically paired with LightGlue[27] or SuperGlue[28]: we opted for the former due to its faster processing time and more permissive licensing. For these methods, we used the implementation available in the deep-image-matching library[4] (DIM) [23], which prepares image matching results directly for the import into COLMAP.

3.2 *Real-time processing*

For real-time collaborative mapping, two distinct approaches were tested. The first is a more photogrammetric approach, referred to as OtF-SfM [20], while the second is based on V-SLAM, specifically CCM-SLAM [22]. The OtF-SfM approach utilizes a server-client system, where clients continuously acquire images and send them to a central server. The server groups the images based on similarity and attempts to orient them into a single map. When not feasible, sub-maps are generated and later merged once a sufficient number of common images are available. The incremental mapping process is based on COLMAP, a new fast image retrieval strategy, and a weighted BA based on image similarity.

For the V-SLAM based approach, we chose CCM-SLAM because it is a state-of-the-art collaborative SLAM system that relies only on visual odometry (VO) without inertial sensor fusion. Every VO agent in CCM-SLAM is built on an ORB-SLAM front-end, currently a widely used monocular SLAM solution that limits the amount of observations using only the richest frames through a keyframe selection procedure. Keyframe poses, keypoints, descriptors, and 3D tie points are sent to the server that builds local maps, attempts to close loops, and merges these maps using place recognition[29] with iterative global bandle adjustments (GBA). The agents download updated keyframe poses from the server after GBA, improving local map accuracy and trajectory estimates with information from other agents.

Since neither OtF-SfM nor CCM currently incorporate GNSS data in trajectory estimation, all results presented here are based solely on image processing. Except for CCM, video frames were selected based on a temporal interval of 1Hz, ensuring approximately 80% overlap between consecutive images. In the case of CCM, however, the frames were recorded to a ROSBAG file at the frame rate of the video itself which was 29.97 Hz (refer to Table 2). Real-time processing evaluation was performed on an Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz with an Nvidia 2060 Rtx GPU.

[4] https://github.com/3DOM-FBK/deep-image-matching/

4 RESULTS

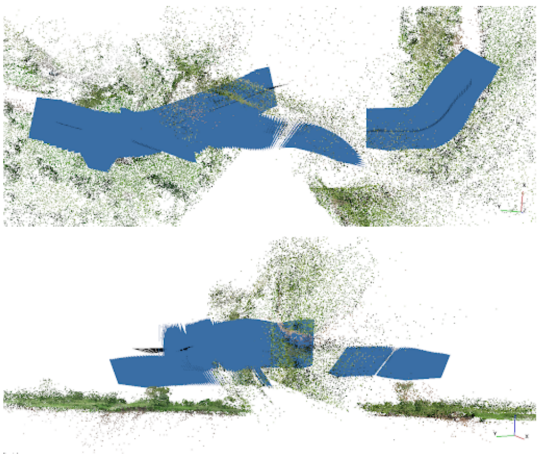
This section presents the processing results for datasets 1 and 2, comparing the estimated trajectories with the GNSS data used as ground truth. RMSE is reported for each agent individually and for the entire dataset when more than one agent trajectories have been estimated together (Tables 4 and 5). The processing with traditional SfM software (Metashape and COLMAP used as reference) highlights the complexity of the analyzed datasets, despite their apparent simplicity. At a semantic level, many savanna characteristic elements are well identifiable, such as the river, trees and bushes. At a more granular level, however, the keypoints are quite poor, since they are extracted on the scale of grass and leaves. In both datasets, Metashape failed to correctly orient the image block, creating degenerate trajectories and maps. This happened probably due to the poor quality of the tie points (Figure 1a and 2a). Also COLMAP with RootSIFT features (default) shows degenerate results for dataset 1 (Figure 1b), while for dataset 2 (Figure 2b) it is the only approach that oriented images from all three agents together with the RMSE of 1.10 m (see last column of Table 5). In general, for all approaches, the major challenge was to orient images from drones flying from the opposite viewpoints. For this reason, COLMAP was tested with the use of SuperPoint, a local feature trained for these kinds of difficult viewing angles. In dataset 1, it succeeds in orienting the agents individually, obtaining the best RMSE (0.20 and 0.11 m), but in dataset 2 it only succeeds in orienting the two agents flying in the same direction and not the third, while COLMAP with RootSIFT succeeded. To understand the possibility of working with variation in flight heights to cater to multiple fieldwork requirements, the results seen so far for SfM processing are all self-calibrated.

Dataset 1	**RMSE [m] on trajectory**	
	Flight 1	Flight 2
Agisoft Metashape	degen	degen
COLMAP	degen	degen
COLMAP (SuperPoint)	0.20	0.11
OtF-SfM	0.42	0.50
OtF-SfM (SuperPoint)	0.45	1.18
CCM-SLAM	0.30	0.26

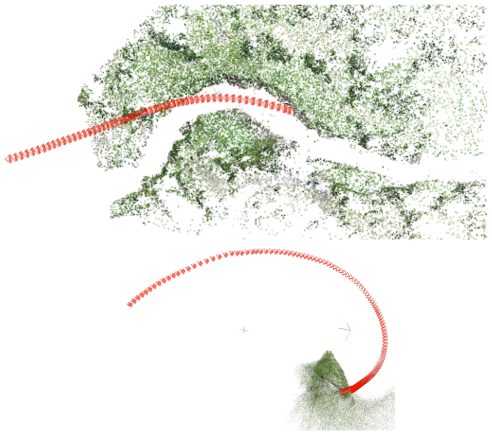
Table 4: Dataset 1 results in terms of RMSE of trajectories compared with GNSS positions. Due to critical difference in viewpoint of agent cameras, collaborative mission simulation results were degenerate in all applied methods, hence only single agent trajectories were compared.

Collaborative approaches did not produce degenerate re-

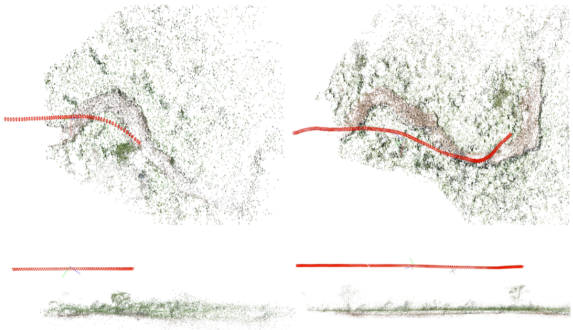
http://www.imavs.org/



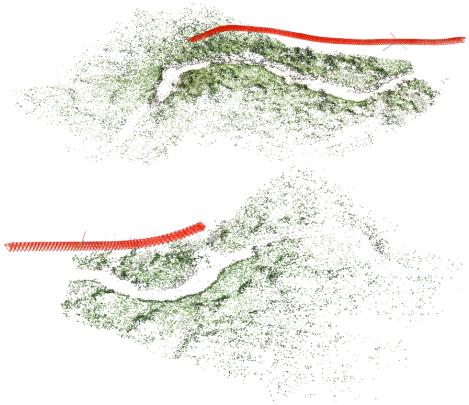
(a) Metashape



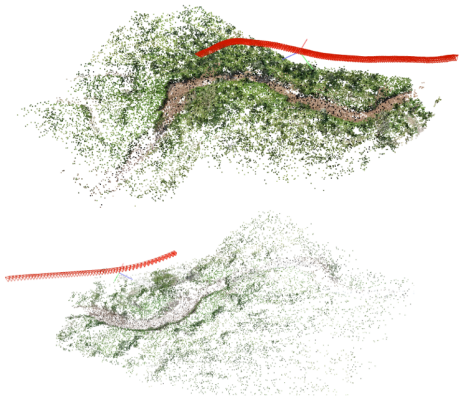
(b) COLMAP



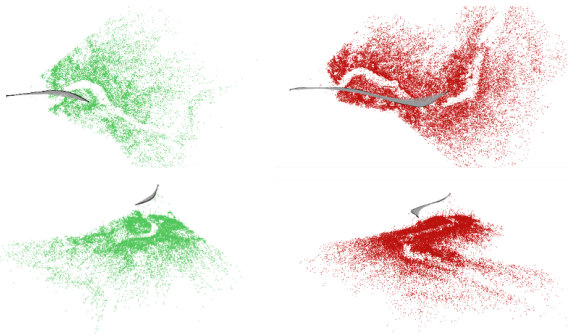
(c) COLMAP + Superpoint



(d) OtF-SfM



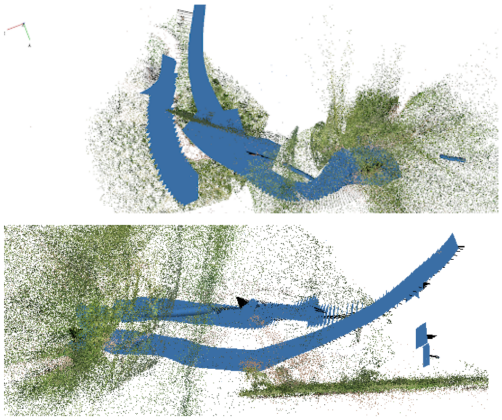
(e) OtF-SFM + SuperPoint



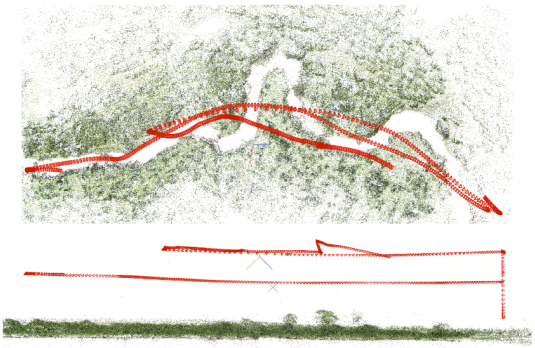
(f) CCM-SLAM

Figure 1: Visualization of image orientation results for dataset 1. (a) Degenerate trajectories from Metashape; (b) Degenerate trajectory recovered for agent 1 in COLMAP, shown from top and side view; (c) Trajectories in in COLMAP (SuperPoint) for both agents, shown fr top and side view; (d) OtF-SFM trajectories for both agents shown from an oblique view point; (e) OtF-SfM (SuperPoint) trajectories for both agents shown from oblique view point; (f) CCM-SLAM trajectories for individual agents shown from top and oblique front view (green point cloud represents mapping through agent 1 and red point cloud from agent 2).

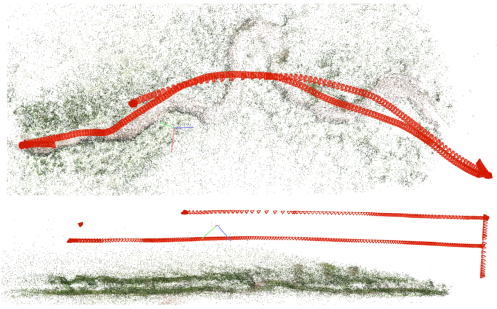
http://www.imavs.org/



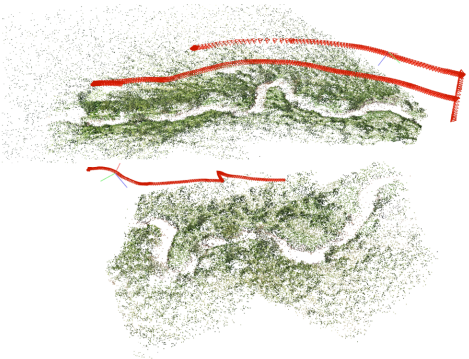
(a) Metashape



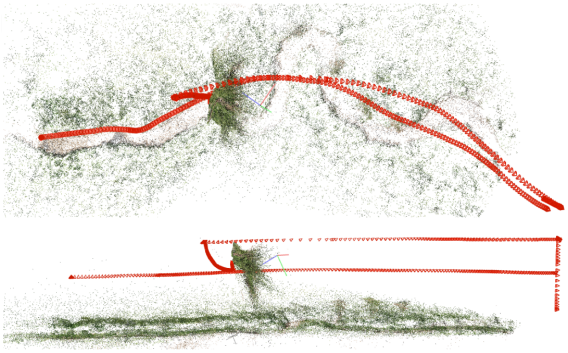
(b) COLMAP



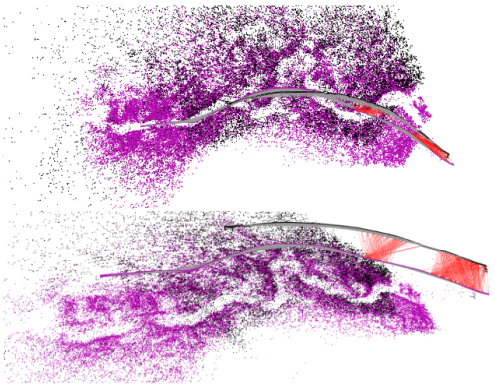
(c) COLMAP + Superpoint



(d) OtF-SfM



(e) OtF-SfM + SuperPoint



(f) CCM-SLAM

Figure 2: Visualization of image orientation results for dataset 2. (a) Degenerate trajectories from Metashape; (b) Trajectories recovered from all three agents in COLMAP shown from nadiral and side view; (c) Trajectories recovered with in COLMAP (SuperPoint) for two agents shown from top and side view; (d) OtF-SfM trajectories for two agents shown from oblique view points; (e) OtF-SfM (SuperPoint) trajectories for two agents shown from nadiral and side view point; (f) Collaborative mapping results in CCM-SLAM for two agents (1 and 3) shown from top and oblique side view (red lines indicate the location matches in key frames).

Dataset 2	RMSE [m] on Trajectory				
	Fl 1	Fl 2	Fl 3	Coll	#
Agisoft Metashape	degen	degen	degen	-	0/3
COLMAP	0.39	0.27	0.30	1.10	3/3
COLMAP (SuperPoint)	0.28	0.27	0.36	1.03	2/3
On-the-Fly	1.52	0.80	0.98	1.54	2/3
On-the-Fly (SuperPoint)	degen	1.51	0.69	1.46	2/3
CCM-SLAM	0.74	0.54	0.81	6.39	2/3

Table 5: Dataset 2 results in terms of trajectories RMSEs. Last column describes the number of agents for which trajectories were recovered during collaborative mission simulation.

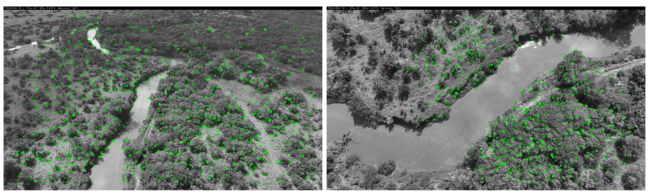


Figure 3: CCM-SLAM loses tracking of features when the camera undergoes rotation movement. RQt visualization shows the reduction in tracked features as the drone turns.

constructions, probably thanks to mandating the use of pre-calibrated cameras, which reduced the influence of outliers and inaccuracies in tie points as they were easily filtered out. Furthermore, contrary to traditional SfM methods, they generated trajectories and maps in real-time, but generally at the cost of higher RMSE (Table 4 and 5). Neither OtF-SfM with SIFT or SuperPoint, nor CCM-SLAM managed to orient agents flying in opposite directions due to the strong change in viewing angle. This problem seems to be related to the performance of the local features and matchers used and the specific implementation, while COLMAP SfM with RootSIFT had been able to orient the entire block. In terms of accuracy, CCM-SLAM was more accurate in dataset 1 than OtF, in particular in the case of using SuperPoint. But in dataset 2, OtF achieved much better accuracy with respect to CCM-SLAM, and with a more complete trajectory. In fact, in CCM-SLAM, upon a critical rotation of agent, tracked features were lost partially (Figure 3) or entirely. This was observed mostly upon fast rotation of camera, swift change in camera pitch or due to small-parallax motion in case of high-altitude UAV flight, such as is performed in wildlife habitat surveys. Since CCM-SLAM cannot recover from lost tracking, parts of the trajectory were lost.

5 CONCLUSIONS

The paper presented results and challenges of UAV-based collaborative 3D mapping for monitoring large-scale savanna environments. The results show that relying solely on visual data is promising, with open-source collaborative frameworks available for real-time processing. However, the solution is prone to the quality of local feature matches. To solve these issues, the addition of IMU or RTK-enabled GNSS data could be considered to avoid mapping errors and degenerate solutions.

When compared to visual-inertial odometry (VIO), purely video based visual odometry (VO) is less common in applications of V-SLAM in outdoor environments [17]. VO can be sensitive to repetitive textures, dynamic scenes and is more prone to tracking losses or initialization failures. Inertial measurements make outdoor operations more robust to challenging situations, compared to VO [30].

However, ensuring that a comparable quality of results is achieved with just visual data, a purely photogrammetric approach has greater potential for widespread use in widelife research. This is due to the popularity and high availability of off-the-shelf drones, which can easily provide high-quality video sequences, but inertial measurements and RTK GNSS positions can be hard or impossible to retrieve from their data logs. Therefore, evaluating the performance of real-time camera-based pipelines such as Visual-SLAM or OtF-SfM is crucial to understand their applicability to UAV swarm technology for wildlife conservation through habitat monitoring.

ACKNOWLEDGEMENTS

The WildDrone project https://wilddrone.eu/ has received funding from the European Union's Horizon Europe research and innovation programme under the Marie Skłodowska-Curie grant agreement no. 101071224.

REFERENCES

[1] D. Tuia, B. Kellenberger, S. Beery, B.R. Costelloe, S. Zuffi, B. Risse, A. Mathis, M.W. Mathis, F. van Langevelde, T. Burghardt, R. Kays, H. Klinck, M. Wikelski, I.D. Couzin, G. van Horn, M. C. Crofoot, C.V. Stewart, and T. Berger-Wolf. Perspectives in machine learning for wildlife conservation. *Nature Communications*, 13(1):792.

[2] A. Wirsing, A. Johnston, and J. Kiszka. The rapidly expanding role of drones as a tool for wildlife research. *Wildlife Research*, 49, 02 2022.

[3] A. Shapiro, K. Anderson, J. Duffy, F. Spina Avino, L. DeBell, and P. Glover-Kapfer. *Conservation Technology Series Issue 5: DRONES FOR CONSERVATION*.

[4] V. Shukla, L. Morelli, F. Remondino, A. Micheli, D. Tuia, and B. Risse. Towards estimation of 3d poses and shapes of animals from oblique drone imagery. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, XLVIII-2-2024:379–386.

http://www.imavs.org/

[5] M. Barbeau, J. Garcia-Alfaro, and E. Kranakis. Research trends in collaborative drones. *Sensors*, 22(9):3321.

[6] H. Wang, J. Tang, Q. Pan, Z. Zhao, and Z. Wang. Multi-uav collaborative reconnaissance based on adaptive particle swarm optimization. In *2023 IEEE Proc (ICUS)*, pages 559–564, 2023.

[7] R. Huang, H. Zhou, T. Liu, and H. Sheng. Multi-UAV collaboration to survey tibetan antelopes in hoh xil. *Drones*, 6(8):196.

[8] R.H. Kabir and K. Lee. Wildlife monitoring using a multi-UAV system with optimal transport theory. *Applied Sciences*, 11(9):4070.

[9] P. Tong, X. Yang, Y. Yang, W. Liu, and P. Wu. Multi-UAV collaborative absolute vision positioning and navigation: A survey and discussion. *Drones*, 7(4):261.

[10] W. Chen, X. Wang, S. Gao, G. Shang, C. Zhou, Z. Li, C. Xu, and K. Hu. Overview of multi-robot collaborative SLAM from the perspective of data fusion. *Machines*, 11:653.

[11] F. Poiesi, A. Locher, P. Chippendale, E. Nocerino, F. Remondino, and L. Van Gool. Cloud-based collaborative 3d reconstruction using smartphones. In *Proc CVMP*, CVMP '17, 2017.

[12] B. Koger, A. Deshpande, J.T. Kerby, J.M. Graving, B.R. Costelloe, and I.D. Couzin. Quantifying the movement, behaviour and environmental context of group-living animals using drones and computer vision. *Journal of Animal Ecology*, 92(7):1357–1371, 2023.

[13] K. Wang, L. Kooistra, R. Pan, W. Wang, and J. Valente. UAV-based simultaneous localization and mapping in outdoor environments: A systematic scoping review. *Journal of Field Robotics*.

[14] A.J. Davison, I.D. Reid, N.D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067.

[15] M.F. Ahmed, K. Masood, V. Fremont, and I. Fantoni. Active SLAM: A review on last decade. *Sensors*, 23(19):8097.

[16] H. Cao, J. Xu, Z. Yang, L. Shangguan, J. Zhang, X. He, and Y. Liu. Scaling up edge-assisted real-time collaborative visual SLAM applications. *IEEE/ACM Transactions on Networking*, 32(2):1823–1838.

[17] Z. Danping, P. Tan, and W. Yu. Collaborative visual SLAM for multiple agents:a brief survey. *Virtual Reality & Intelligent Hardware*, 1:461–482.

[18] S. Jiang, C. Jiang, and W. Jiang. Efficient structure from motion for large-scale UAV images: A review and a comparison of SfM tools. *ISPRS Journal of Photogrammetry and Remote Sensing*, 167:230–251.

[19] V.K. Reja, K. Varghese, and Q.P. Ha. Computer vision-based construction progress monitoring. *Automation in Construction*, 138:104245.

[20] Z. Zhan, R. Xia, Y. Yu, Y. Xu, and X. Wang. On-the-fly SfM: What you capture is what you get. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, X-1-2024:297–304.

[21] Y. Yue, C. Zhao, Y. Wang, Y. Yang, and D. Wang. Aerial-ground robots collaborative 3d mapping in gnss-denied environments. In *Proc. ICRA*, pages 10041–10047, 2022.

[22] P. Schmuck and M. Chli. CCM-SLAM: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams. *Journal of Field Robotics*, 36(4):763–781.

[23] L. Morelli, F. Ioli, F. Maiwald, G. Mazzacca, F. Menna, and F. Remondino. Deep-image-matching: a toolbox for multiview image matching of complex scenarios. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, 48:309–316, 2024.

[24] J.L. Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proc CVPR*, pages 4104–4113, 2016.

[25] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *2012 IEEE Proc CVPR*, pages 2911–2918. IEEE, 2012.

[26] D. DeTone, T. Malisiewicz, and A. Rabinovich. SuperPoint: Self-supervised interest point detection and description. In *Proc. CVPR 2018*, pages 337–33712.

[27] P. Lindenberger, P.-E. Sarlin, and M. Pollefeys. LightGlue: Local feature matching at light speed. In *Proc. ICCV 2023*, pages 17581–17592.

[28] Paul-Edouard Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *Proc. CVPR 2020*, pages 4937–4946.

[29] D. Galvez-López and J.D. Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197.

[30] J. Cremona, R. Comelli, and T. Pire. Experimental evaluation of visual-inertial odometry systems for arable farming. *Journal of Field Robotics*, 39(7):1121–1135.

http://www.imavs.org/

Effects of Wind Speed on Quadcopter Ceiling Effect

Rhiannon Elliott-Roe*, Dr Kieran Wood, Dr Ozgun Ozer
University of Manchester, Oxford Road, Manchester

http://www.imavs.org/

ABSTRACT

The ceiling effect is an aerodynamic condition observed when a rotorcraft is in proximity of a horizontal, rigid surface above the rotors. It is in some ways like the ground effect, which has been well researched over the years both in the context of conventional helicopters and more modern quadcopters. This paper aims to investigate the ceiling effect and its impact on quadcopters in proximity flight, with a specific focus on whether there is a quantifiable difference in the thrust produced due to the ceiling effect in wind versus still air.

NOMENCLATURE

α_0, α_1	Dimensionless empirical constants which describe nonaxisymmetric flow and wake re-circulation.
T_{ICE}	Thrust in ceiling effect
T_{OCE}	Thrust out of ceiling effect
R	Radius of propeller
Z	Distance between propeller plane and artificial ceiling
$\hat{z}$	Z/R

1 INTRODUCTION

Unmanned Aerial Vehicles (UAVs) have been steadily growing in popularity since the mid-2010's as a tool for scientific research [1]. UAVs are typically limited by the battery capacity or the amount of fuel it can carry onboard, which can make it difficult to retrieve data from remote, inaccessible locations. Aerial rendezvous has the potential to increase the time-on-task for a data-gathering UAV by ferrying the UAV to its desired location via a larger parent vehicle with better endurance characteristics. The data-gathering vehicle in this case is a quadcopter, and the parent vehicle a fixed wing UAV. Once the data-gathering vehicle has performed its task, it first would then rendezvous with the parent UAV, which will take it back to the desired landing location. Addressing this problem requires an investigation into the disturbance effects of two vehicles flying in proximity.

*Email address: rhiannon.elliott@manchester.ac.uk

One of these proximity disturbances is known as the ceiling effect, which can be observed when the inflow through the propeller is obstructed to some degree by an overhead surface, which decreases the induced velocity across the rotor disk. This increases the thrust generated by the rotor, so it appears as if the quadcopter is being sucked up towards the overhead surface. In the context of aerial rendezvous, it is expected that a quadcopter could be forced into the wing of the parent vehicle, destabilising one or both UAVs. Evidently, this has the potential to cause catastrophic damage to the vehicle, the ceiling and any operators nearby, consequently, recent contributions to flight controller design have sought to sense and avoid the ceiling effect at all costs [2].

There is a strong need for research in proximity aerodynamics on small UAVs to aid the generation of mathematical models [3, 4]. These mathematical models are paramount for controller development and therefore reliable and predictable autonomous flight. Fully autonomous flight forms the foundation of operations which require faster responses than can be expected from a remote pilot; for example when flying in and around other UAVs or attempting complex manoeuvres such as mid-air rendezvous. As aircraft rendezvous is a complex and risk intensive process, the problem must be simplified. This may be done by modelling the underside of the wing as an infinite ceiling. This assumption would be appropriate when considering a small fixed wing quadcopter and a sufficiently large fixed wing UAV.

This paper aims to investigate the ceiling effect and its impact on thrust produced by a propulsion unit in proximity to an infinite ceiling. The focus of the experiment is to determine if wind speed causes a quantifiable difference in the thrust produced via ceiling effect. This will be done by first validating the experimental procedure against published theoretical models using a single rotor test case with no wind present. The test will then be repeated at different wind speeds.

2 LITERATURE REVIEW

One of the key examples of characterisation of rotor proximity interactions dates back to 1957, and discusses helicopters and the ground effect observed by pilots when coming into land. Although the idea of the ceiling effect was discussed, there was no interest to pursue this research due to the lack of operational requirement for helicopters to fly within such proximity to a rigid, overhead barrier. As UAV

technology develops, there is increasing interest in flight controllers that can safely navigate near large, rigid bodies [5].

"The Effect of the Ground on a Helicopter Rotor in Forward Flight" by Cheeseman and Bennett [6] focuses on the ground effect, an aerodynamic event that occurs when a helicopter approaches the ground and experiences an increase in thrust due to a cushion of air between the rotor plane and the ground. Cheeseman and Bennet put forward an approximate theory to estimate the thrust generated within ground effect, and how it changed with increasing forward airspeed. The findings from this paper suggested that the relative thrust increases as the distance between the rotor plane and the ground decreases, and that increasing forward speed decreases the relative thrust. This paper remains significant today and is featured in several pieces of literature that has been published in the last few years.

D Carter et al aimed to improve the design of near boundary flight controllers by investigating existing models for ground, ceiling and wall effects in *"Influence of the Ground, Ceiling and Sidewall on Micro-Quadcopters"* [7]. Through experimental testing, they were able to deduce that the theoretical models often over-predicted the thrust increase caused by the ceiling effect at small distances from the ceiling. This paper was of particular significance due to its Particle Image Velocimetry (PIV) studies which clearly illustrated the flow state around the propeller plane when in ceiling effect. Both the experimental validation of theoretical models, as well as PIV studies showing the complex inflow characteristics, have enriched the field of literature and gone on to aid further research on the ceiling effect.

In *"Multi-rotor UAS for Bridge Inspection by Contact using the Ceiling Effect"*, P Sanchez-Cuevas et al set out to capitalise on the ceiling effect by designing a protective fairing to shield the propellers on a quadcopter while maintaining the optimum distance between the ceiling and the rotor plane [5]. The optimum distance was determined using experimental methods like those discussed in [7] and was successfully validated by demonstrating the potential use as a bridge surveying tool. The quadcopter was able to remain attached to the ceiling at a reduced power setting as a direct result of the ceiling effect.

As perhaps one of the most significant works on ceiling effect, *"Ceiling Effects for Hybrid Aerial Surface Locomotion of Small Rotorcraft"* written by Hsiao et al explored the ceiling effect as a power conserving strategy [8]. By using Momentum Theory and Blade Element Method to complete a robust theoretical analysis of the phenomena, they were able to successfully generate a model for a quadrotor in ceiling effect as a function of blade radius, distance from the ceiling and an empirical recirculation factor. While their model was verified using experimental testing, there are however limitations to the applications for other drones as the empirical constants would need to be determined for each drone type.

"An Empirical Evaluation of Ground, Ceiling and Wall Effect for a Small-Scale Rotorcraft" written by S Conyers et al; brings unique attention to the demands of the drone community for flying quadcopters indoors, and specifically that more work must be done to design flight controllers to fly within these complex flight regions rather than avoiding them entirely [2]. The authors propose that the ceiling obstructs the intake of the rotors, reducing the induced velocity of the flow and therefore increasing the thrust produced by the rotors for a given power setting. This is due to the inverse proportionality between thrust from the rotor and the induced velocity across the rotor disk, which can be shown when applying the conservation of momentum principle from far upstream of the propeller to far downstream. In addition to this, they address the uncertainty surrounding the validity of ceiling effect models that were initially based on ground effect models for single rotor helicopters. They suggest that the ceiling effect model can be applied to multi-rotors as the ceiling effect is an upstream phenomenon, whereas rotor wake interactions are a downstream phenomenon that should not interact with the ceiling effect to a significant extent.

Overall this section shows the accelerated pace at which research into the ceiling effect and how it interacts with quadcopters is being carried out. There is a noticeable shift in recent years where researchers aim to design flight controllers that can contend with the hazardous near-boundary flow of the ceiling effect, rather than minimising the time spent by the quadcopter in those regions. This paper aims to contribute to this research field by exploring how the ceiling effect may vary with wind speed. The findings from this research will support the development of proximity flight controllers which is a significant stepping stone along the path to reliable, autonomous aerial rendezvous.

3 THEORY

A low order modelling approach is adopted here to predict the ceiling effect in non-stationary conditions. The Hsiao and Chirarattananon model described in [8] and shown in 1, contains two empirically derived, dimensionless constants, α_0 and α_1, to describe the non-axisymmetric flow and wake circulation respectively. Since this experiment will only consider a flat ceiling with respect to the rotor plane, the constant α_0 can be set to 1. Hsiao et al stated that where α_1 is assumed to be zero, for $\hat{z} > 0.05$ the model is still sufficiently accurate. Hence, substituting α_0 and α_1 to 1 and 0 respectively gives Equation 2.

http://www.imavs.org/

$$\frac{T_{ICE}}{T_{OCE}} = \frac{1}{2}\left(1 - \frac{\alpha_1}{\hat{z}^2}\right) + \frac{1}{2}\sqrt{\left(1 - \frac{\alpha_1}{\hat{z}^2}\right)^2 + \frac{\alpha_0}{8\hat{z}^2}} \quad (1)$$

$$\frac{T_{ICE}}{T_{OCE}} = \frac{1}{2} + \frac{1}{2}\sqrt{1 + \frac{1}{8\hat{z}^2}} \quad (2)$$

Where $\hat{z} = Z/R$

Equation 1 has been verified experimentally for single rotor and quadcopter tests in [7, 9] and was found to have good agreement with theoretical data for the case where α_0 = 1 and α_1 = 0.

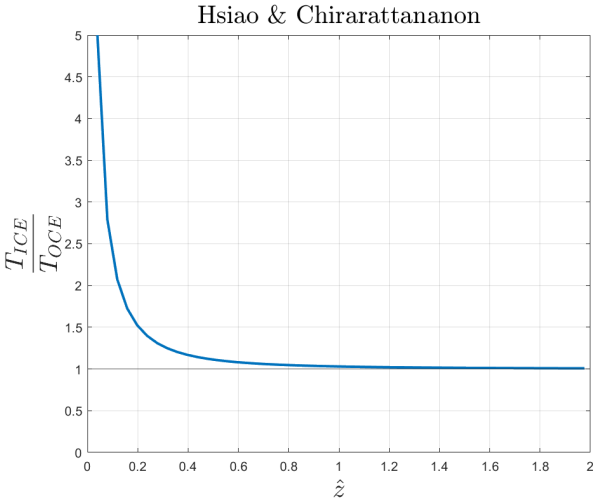


Figure 1: Hsiao and Chirarattananon Theoretical Model [8]

4 EXPERIMENTAL PROCEDURE

This section details the full experimental procedure for this paper including the apparatus, methodologies for both experiments and the data processing method. Initially, a single rotor test is carried out to verify the experimental data gathered agrees with the theoretical models published in the literature. The quadcopter data will be compared against the single rotor test as well as the theoretical model.

4.1 Apparatus

4.1.1 Wind Tunnel

This experiment was conducted in the open-circuit project wind tunnel at the University of Manchester, which can be seen in Figure 2. The test section is 1.2 x 0.9 x 2m and has a maximum speed of 50m/s [10] as shown in Figure 2. Within the wind tunnel, the boundary layer forms before the test section begins. In the worst case scenario the boundary layer would be approximately 8.71mm according to the Blasius equation as seen in equation 2, and using the lowest wind speed of interest 10m/s [11]. To avoid interaction with the true ceiling boundary layer, the false ceiling will be suspended at least 50mm below the true ceiling to minimise the risk of boundary layer interaction from the true ceiling of the wind tunnel. The boundary layer formed on the artificial ceiling will be approximately 5.84mm thick at its maximum point which is acceptable as the closest the propeller will be to the ceiling is 15mm away.

$$\delta_{99}(x) \approx 5.0\sqrt{\frac{\nu x}{u_0}} = 5.0\frac{x}{\sqrt{Re_x}} \quad (3)$$



Figure 2: Project Wind Tunnel at the University of Manchester

4.1.2 Load Cell

A six-axis load cell (Mini40-E, ATI Industrial Automation, USA)[12] was used to collect measurements of the force and torques acting on the rotor and complete vehicle. The load cell is calibrated over a small range appropriate for the the up to 20 N forces predicted by theory.

4.1.3 Propulsion Unit

The experiment was conducted using an isolated propulsion unit. This unit is defined here as the entire power train required to convert electrical energy into useful work, hence; propeller (Holybro 1045), motor (Holybro 2216 920KV), and ESC (X500 V2 BLHeli S 20A). Power is provided by a 50 A bench supply to ensure experiments have a constant voltage input of 12V. A commutation RPM sensor was added (Hobbywing RPM Sensor) [13] between the motor and ESC, which was connected to a SpeedyBee F405 WING APP Fixed Wing Flight Controller [14] that recorded the rpm of the motor in the data logs.

http://www.imavs.org/

http://www.imavs.org/

4.1.4 Motor stand

A telescopic speaker stand was used to vary the height between the artificial ceiling and the rotor plane. Adaptors were 3D printed using PLA so the propulsion unit could be fixed to the load cell, on top of the stand. The stand was required to move the rotor plane from 2 to 30cm away from the artificial ceiling, the minimum distance was set to 2cm to reduce the risk of the propeller blade striking the artificial ceiling. It was necessary to measure the distance between the rotor plane and the artificial ceiling without opening the wind tunnel. To do this, the motor stand was raised so that the rotor hub was just touching the artificial ceiling. From underneath the wind tunnel, the motor stand was marked where the bottom of the wind tunnel intersected with the stand. This point was marked as 0cm. The remaining markers were added from 2 to 30cm so that value intersecting the bottom of the wind tunnel represented the distance between the propeller and the ceiling. This allowed the distance between the rotor plane and the artificial ceiling to be adjusted without the need to open the wind tunnel. To prevent the speaker stand from moving during the experiment, slotted weights were used to weigh down the bottom of the tripod.

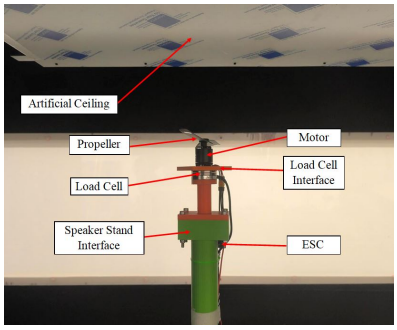


Figure 3: Annotated Picture of Single Motor on Test Stand

5 METHODOLOGY

The motor was controlled using a SpeedyBee F405 WING APP Fixed Wing Flight Controller [14] and the open-source program Mission Planner by ArduPilot. In Mission Planner, the motor test function allows the user to set the desired throttle percentage and test length in seconds. The RPM of the propeller was recorded using a commutation RPM sensor (Hobbywing RPM Sensor, [13]), with data being logged using the SpeedyBee Flight Controller. RPM was measured to verify that the propeller RPM was constant throughout the test, which was a key assumption in the theory set out by Hsiao et al [8].

The rotor axis was aligned with the z-axis of the load cell, so the force recorded by the load cell is equivalent to the thrust generated by the propulsion unit. The experiment was started with no wind and the wind tunnel closed. Each data point taken represents five, five seconds long motor tests at a given throttle setting and distance. The distance between the propeller and the artificial ceiling was gradually increased from 2 to 30cm, in 1cm increments. Once the thrust and RPM data was recorded for each distance and throttle setting, the wind tunnel was turned on and the experiment was repeated at 5, 10 and 15m/s wind speed.

5.1 Data Processing

The load cell data was recorded directly using a LabView VI at a sampling rate of 10,000Hz. This data was then processed in MATLAB by using a 1D moving average filter, with a window size of 250 points to get an average thrust reading for each test. The average thrust reading was normalised by the Out-of-Ceiling effect thrust value which was taken at the maximum distance from the ceiling.

The RPM data was logged on the flight controller, then extracted and converted to a MATLAB file using Mission Planner. The mean RPM and the standard deviation was then calculated across each wind speed and throttle setting.

6 RESULTS

A range of throttle settings were used to investigate whether the normalised thrust was dependent on the throttle setting. The results at 0m/s are shown in Figure 4; each solid curve represents the fitted data at each wind speed and throttle setting. Figure 4 shows that there is no particular pattern which indicates throttle setting effects the normalised thrust produced. The same process was carried out at each subsequent wind speed, which also showed no link between throttle setting and thrust produced.

A key assumption in the theory described by Hsiao et al [8], was that RPM must remain constant at each distance away from the ceiling . In Table 1, the mean RPM and and standard deviation as a percentage of RPM is shown. Overall, the standard deviation is low across the entire data set, which suggests that on the whole the RPM was maintained well. The larger standard deviations occur at wind speeds of 10 and 15 m/s, with the greatest deviation of 1.28% occurring at 55% throttle and 10m/s. It was observed during wind speeds of 10 and 15 m/s, that the propeller appeared to rock on the motor shaft. This rocking may cause inconsistencies with the RPM and would likely explain the greater deviation from the mean at higher wind speeds.

The propulsion unit was tested at 0, 5, 10 and 15 m/s wind speeds, and at 55%, 65% and 75% throttle, the results are shown in Figures 5, 6 and 7. The figures suggest that the normalised thrust data appears to follow the curve of theoretical thrust values (depicted in the figures with the

	Throttle Setting [%]					
	55		65		75	
Wind Speed [m/s]	Mean RPM	Relative Standard Error	Mean RPM	Relative Standard Error	Mean RPM	Relative Standard Error
0	8043.067647	0.53%	9109.16	0.41%	10048.63	1%
5	8140.434532	0.42%	9235.253	0.46%	10207.88	1%
10	8125.405918	1.28%	9189.042	1.24%	10158.74	1%
15	8125.709903	1.27%	9189.31	1.24%	10158.89	1%

Table 1: Mean and Standard Deviation of RPM

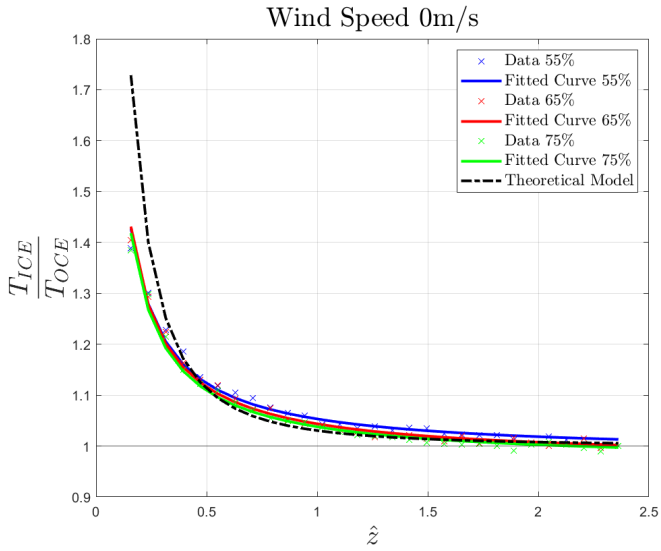


Figure 4: Normalised Thrust vs Normalised Distance From Ceiling for Propulsion Unit in 0m/s Wind

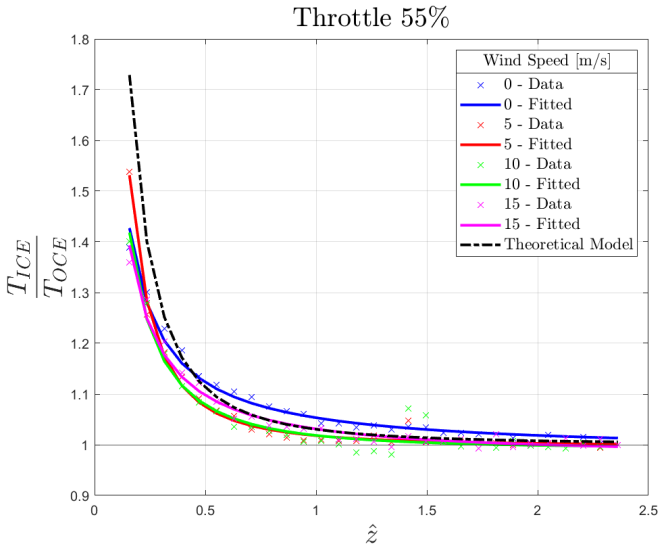


Figure 5: Normalised Thrust vs Normalised Distance From Ceiling for Propulsion Unit at 55% Throttle

dashed black line) as described by Equation 1 to some extent. However at normalised distances of 0.5 and less, the normalised thrust values appear to be less than that predicted by Equation 1.

To quantitatively evaluate the fit of these curves, the Mean Absolute Error for each data set was calculated, and is expressed in Table 6 as a percentage of the range of each data set. Overall, the Mean Absolute Error for each fit ranges from 3.689% to 8.226% which indicates that the data is loosely described by Equation 2, which was used as the model to fit the data against. The discrepancies between the experimental data and theoretical values at normalised distances less than 0.5 may explain why the Mean Absolute Error is as large as it is. There is no obvious trend as to whether the goodness of fit increases with either wind speed or throttle setting.

Figures 5, 6 and 7 show for each throttle setting, the normalised thrust at each different wind speed. If increasing the wind speed decreased the thrust produced via ceiling effect, this would be shown by the curve shifting downwards on the y axis with increasing wind speed. This is not the case in these figures, as there is no consistent trend between the wind speed and normalised thrust produced in either Figure 5, 6 or 7. The figures and goodness of fit measurements shown suggest that wind speed does not cause an quantifiable difference in the thrust produced via ceiling effect for this specific configuration of propulsion unit and artificial ceiling.

7 DISCUSSION AND CONCLUDING REMARKS

This investigation into the ceiling effect and wind speed shows that with this configuration of propulsion unit and artificial ceiling, there is no quantifiable change in the thrust produced with varying wind speed. The normalised thrust values recorded during this experiment at 0m/s, are in good agreement with the results of [7, 8]. Notably how experimental values of normalised thrust are less than those predicted by the theory when $\hat{z}$ approaches zero.

http://www.imavs.org/

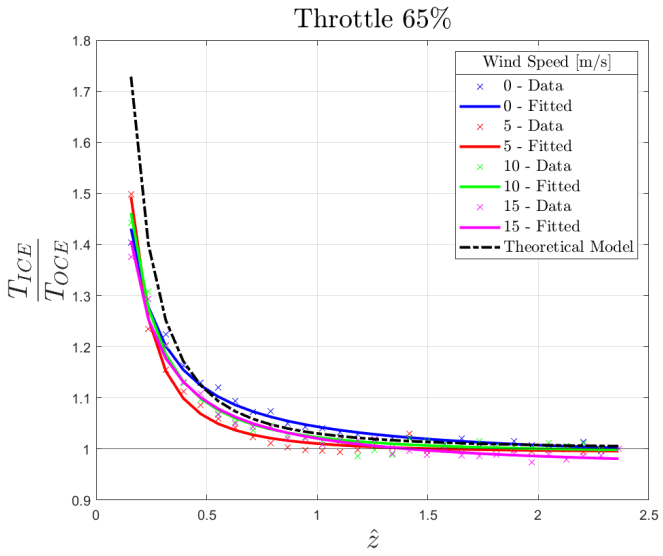


Figure 6: Normalised Thrust vs Normalised Distance From Ceiling for Propulsion Unit at 65% Throttle

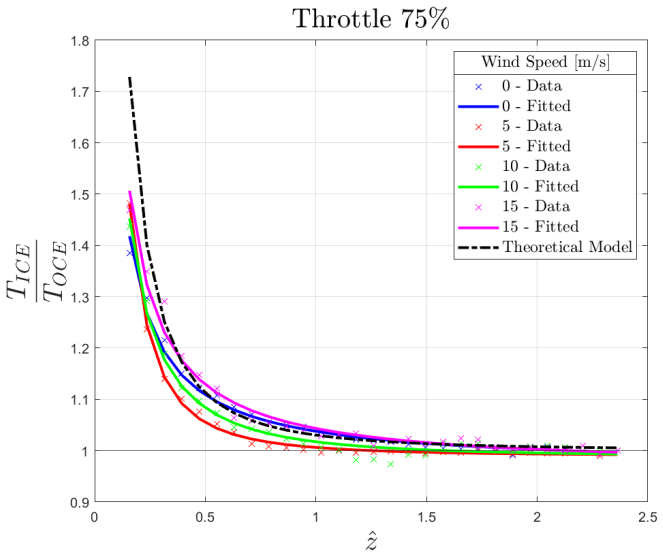


Figure 7: Normalised Thrust vs Normalised Distance From Ceiling for Propulsion Unit at 75% Throttle

	Mean Absolute Error as % of Range		
Wind Speed [m/s]	Throttle 55%	Throttle 65%	Throttle 75%
0	7.297	5.279	5.371
5	4.806	7.077	8.226
10	7.572	5.338	6.633
15	6.293	8.054	3.689

Table 2: Mean Absolute Error Expressed as a Percentage of the Range

This suggests that the experimental procedure in the wind tunnel is valid for measuring the ceiling effect. The experimental method was not sufficient for ruling out whether wind speed has a quantifiable effect on ceiling effect for all cases. The phenomenon described by Cheeseman and Bennett in [6] which suggested that the ground effect decreased with increasing was in reference to helicopters which can fly much faster than a typical quadcopter. If these experiments were repeated at comparable wind speeds there may be a more obvious trend in the effect of wind speed on ceiling effect that mirror the results seen in [6].

These results are useful to the growing community of researchers exploring aerial rendezvous and proximity flight disturbances on UAVs [15]. Modelling the ceiling effect is key to developing controllers capable of anticipating aerodynamic disturbances such as the increase in thrust caused by the ceiling effect. In this specific case, the lack of relationship between the thrust increase due to ceiling effect and wind speed, means wind speed can effectively be ignored when modelling this specific proximity flight disturbance. A less complex model may then make it easier to develop a controller for flying within proximity of overhead objects.

Further work must be done to address the limitations of this experimental model. The assumption that the propeller is parallel to the overhead ceiling would not hold in the real life situation where a quadcopter is flying at the speeds considered in this experiment, as quadcopters must have a non-zero pitch angle to move laterally. There are some examples in the literature of ceiling effect studies against non-parallel surfaces [16, 7, 4]. However, as none consider wind speed, this may be the next logical step in investigating whether wind speed has a quantifiable effect on the thrust produced in the cieling effect.

REFERENCES

[1] Anthony Harrington. Who Controls the Drones? *Engineering and Technology Magazine*, pages 80–83, 2015.

[2] Stephen A. Conyers, Matthew J. Rutherford, and Kimon P. Valavanis. An Empirical Evaluation of Ground Effect for Small-Scale Rotorcraft. pages 1244–1250, 2018.

[3] Derrick W Yeo, Nitin Sydney, Derek A Paley, and Donald Sofge. Downwash Detection and Avoidance with Small Quadrotor Helicopters. Technical report.

[4] G. Throneberry, C. M. Hocut, and A. Abdelkefi. Multirotor wake propagation and flow development modeling: A review, 11 2021.

http://www.imavs.org/

[5] P J Sanchez-Cuevas, Guillermo Heredia, and Anibal Ollero. *Multirotor UAS for Bridge Inspection by Contact Using the Ceiling Effect*. 2017.

[6] I C Cheeseman and W E Bennett. The Effect of the Ground on a Helicopter Rotor in Forward Flight. Technical report, Ministry of Supply, Aeronautical Research Council Reports and Memoranda, London: Her Majesty's Stationery Office, 1957.

[7] Darius J. Carter, Lauren Bouchard, and Daniel B. Quinn. Influence of the ground, ceiling, and sidewall on micro-quadrotors. *AIAA Journal*, 59(4):1398–1405, 2021.

[8] Yi Hsuan Hsiao and Pakpong Chirarattananon. Ceiling Effects for Hybrid Aerial-Surface Locomotion of Small Rotorcraft. *IEEE/ASME Transactions on Mechatronics*, 24(5):2316–2327, 10 2019.

[9] Basaran Bahadir Kocer, Tegoeh Tjahjowidodo, and Gerald Gim Lee Seet. Centralized predictive ceiling interaction control of quadrotor VTOL UAV. *Aerospace Science and Technology*, 76:455–465, 5 2018.

[10] University of Manchester. Department of Mechanical, Aerospace and Civil Engineering Facilities, Wind Tunnels. https://www.mace.manchester.ac.uk/research/facilities/wind-tunnels/.

[11] Marco Coderoni, Pankaj Rajput, Kalyan Goparaju, and Karthik Remella. Ansys Innovation Courses, Flat-Plate Blasius Solution - Lesson 4, 8 2020. https://courses.ansys.com/index.php/courses/laminar-boundary-layer-theory/lessons/flat-plate-blasius-solution-ansys-innovation-courses/.

[12] ATI Industrial Automation. ATI F/T Mini40-E Load Cell. https://www.ati-ia.com/products/ft/ft_models.aspx?id=mini40.

[13] HobbyWing USA. Hobbywing rpm sensor for high voltage esc. https://www.hobbywingdirect.com/products/rpm-sensor?srsltid=AfmBOoocmldaWX46ehGaHFCp_QnILWY8lNIsJFy2TQ33ClLXFCsWGE-S.

[14] SpeedyBee. SpeedyBee F405 WING APP. https://www.speedybee.com/speedybee-f405-wing-app-fixed-wing-flight-controller/.

[15] Sydney, Nitin and Smyth, Brendan and Paley, Derek A. Dynamic Control of Autonomous Quadrotor Flight in an Estimated Wind Field. pages 3609–3616, 2013.

[16] Yuying Zou, Haotian Li, Yunfan Ren, Wei Xu, Yihang Li, Yixi Cai, Shenji Zhou, and Fu Zhang. Perch a quadrotor on planes by the ceiling effect.

http://www.imavs.org/

Drone Swarms for Animal Monitoring: A Method for Collecting High-Quality Multi-Perspective Data

Edouard G. A. Rolland[*], Kasper A. R. Grøntved[*], Lucie Laporte-Devylder[†], Jenna M. Kline[‡], Ulrik P. S. Lundquist[*], and Anders Lyhne Christensen[*]

http://www.imavs.org/

ABSTRACT

Drone swarms offer great potential for wildlife monitoring, but their real-world use is still limited. This paper addresses the challenge of deploying drones to collect high-quality, multi-perspective data over herds of gregarious animals. We formalise this problem using the novel concept of *surface of interest*, combined with a Lambertian-inspired modelling approach. Together, these elements allow us to create an objective function for data quality that also considers the swarm's impact on animal welfare. Using a centralised controller and particle swarm optimisation, our approach determines the drone configurations that maximise this function. Experiments based on real-world animal spatial distributions show that our algorithm effectively identifies these configurations, paving the way for future field tests.

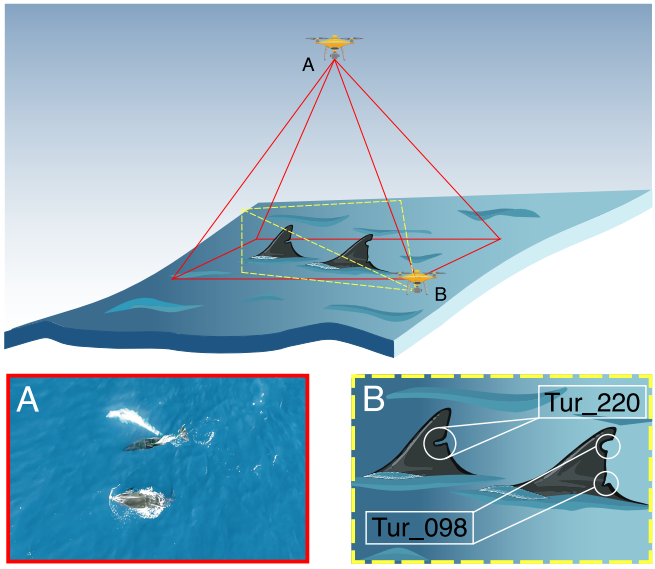


Figure 1: Drones performing (A) vertical and (B) horizontal monitoring of a pod of Tursiops

1 INTRODUCTION

Recently, the alarming decline in biodiversity has initiated the development of technological solutions for conservation efforts in nature [1, 2, 3]. Gathering biological data about endangered species is crucial for tracking the evolution of individuals and understanding how climate change and human activities affect their survival and reproduction. In this context, the recent democratisation of drones has transformed them into a versatile platform for wildlife monitoring surveys [4]. Their cost-effectiveness, time efficiency, and reduced labour demands have established drones as a preferred tool for such work. The literature already highlights numerous drone applications in this field, contributing to significant research output; see [5, 6] for examples.

Challenges persist in conducting effective data collection over large areas, particularly with animals that gather in large groups, commonly known as *gregarious species*. Biologists often rely on a single commercially available multi-rotor drone, operated manually [7]. However, this approach has several limitations. The restricted flight time of the UAV and the limitation of a single vantage point both constrain effectiveness. Additionally, when using non-nadir perspectives, the animals may occlude one another from the drone's view, compromising data collection. Thus, coordinating multiple drones to collect data simultaneously from different angles can address these limitations and introduces new possibilities for biological studies. As illustrated in Figure 1, vertical drone views and horizontal perspectives complement each other well in the context of wildlife monitoring. Vertical monitoring provides a contextual bird's-eye view that offers information about the group and its environment, making it particularly suitable for tasks such as wildlife censuses, movement analysis, and studies of social interactions [2, 8, 9]. In contrast, horizontal monitoring facilitates the integration of individual identification by recognising distinctive marks and patterns on animals [10, 11]. It has been demonstrated that utilising multiple viewpoints and oblique views is more effective for identification than relying solely on a single-view perspective [12]. Multi-perspective data allows for a comprehensive understanding of wildlife dynamics by integrating a

[*]UAS Drone Center, MMMI, University of Southern Denmark (SDU), Campusvej 55, DK-5230 Odense M, {edr,kang,ups,andc}@mmmi.sdu.dk

[†]WIPSEA, Regus, 2 Rue Claude Chappe, 35510 Cesson-Sévigné, France and Department of Biology, University of Southern Denmark (SDU), Campusvej 55, DK-5230 Odense M, lucie.laporte-devylder@wipsea.com

[‡]Department of Computer Science and Engineering, The Ohio State University, 2015 Neil Ave, Columbus, OH, USA 43201, kline.377@osu.edu

http://www.imavs.org/

broad ecological context with precise individual tracking [4].

To our knowledge, only a few studies have focused on the real-world deployment of autonomous drone swarms for wildlife monitoring. Developing such a system with an application-oriented design would enhance the data collection possibilities for biologists and transition drone swarms from simulations and laboratories to real-world applications. Our study consists of three contributions: Firstly, we create a methodology for assessing the quality of multi-perspective monitoring of gregarious animals, considering constraints to avoid disturbing wildlife [13]. Our methodology is inspired by the concept of *Lambertian surfaces*, commonly used in video games to model light reflection [14]. Secondly, we introduce a method for coordinating swarms of drones to maximise the quality of the data collected based on the concept of *surface of interest* (SI). Finally, we evaluate the performance of our solution using a custom-built 3D simulator. The relevance of the controller and its application to real-world scenarios are assessed using various spatial animal distributions based on real-world data.

2 RELATED WORK

Common wildlife monitoring methods, such as GPS trackers, camera traps, and direct field observation, have long been the foundation of ecological research. While they provide valuable data on animal movement, population dynamics, and behaviour, they often struggle to capture comprehensive, multi-perspective data on animals in their natural habitats [7]. Recent advancements in remote sensing technology have introduced novel techniques for conducting animal ecology studies, including drones, acoustic sensors, and satellites [15, 16]. Simultaneously, advancements in machine learning have enabled the analysis of vast volumes of data generated by these remote sensing devices, allowing for the rapid production of ecological studies [17].

Drones are particularly well-suited for collecting ecological data on gregarious animals and have been employed to study a wide range of habitats across various animal species, including Plains and Grevy's zebras, gelada monkeys, giraffes, sheep, and horses [7, 18, 8]. Drones can rapidly traverse remote terrain to track groups of animals over time and offer capabilities often beyond those of camera traps or satellites [16]. Drones have the potential to provide detailed data on all individual animals in a group simultaneously, something that traditional methods cannot capture [19]. In addition to movement, behaviour, and identification in ecological studies, drones have been proposed for 3D pose estimation to evaluate the morphometric parameters of the animals [20, 21].

Techniques for gathering aerial imagery with drones in animal ecology studies vary widely depending on the species and the study's goals. However, they mostly involve collecting information about specific parts of the animals, referred to as SIs. Drones may be piloted to capture either oblique or nadir imagery. In vertical monitoring, the SI typically includes the animal's back. Horizontal monitoring focuses on surfaces where distinctive features for individual identification or fine-grained behaviours are visible, see Figure 2 for examples. Multi-perspective datasets containing both vertical and horizontal monitoring imagery are crucial for comprehensive studies, as they capture behaviour, movement, and identification [22, 8]. Currently, capturing both single- and multi-perspective drone imagery simultaneously is largely manual, requiring close coordination between expert drone pilots and biologists to collect the data while ensuring the animals are not disturbed. This manual approach involves logistical challenges, such as managing drone missions while capturing images [22] and synchronising efforts between operators and biologists [8]. Current protocols for conducting drone-based animal ecology missions are qualitative and largely involve executing missions manually. These protocols do not establish a standard procedure for positioning the drone relative to the SI being studied [22, 10]. Instead, the distance between the animals and the drone is chosen to ensure sufficient resolution for identifying physical characteristics while minimising impact on animal welfare [10].

Autonomous tracking methodologies have been proposed using single drones for cows [23] and yaks [24], and multi-drone swarms for zebras [25]. However, these approaches are typically not generalisable, as they focus on specific species or surfaces of interest and may not address the need to minimise disturbance to the animals. The potential of drone swarms for vertical animal observations has been explored with gregarious species in livestock monitoring [26]. For instance, Li et al. [27] employed density-based clustering to optimise drone deployment, maximising animal coverage while minimising the average distance between drones and animals to ensure high-quality imagery. More broadly, the challenge of covering the maximum number of targets with the minimum number of drones has been addressed using bio-inspired algorithms, for example, the elephant herding optimisation algorithm [28] and artificial bee colony algorithms [29].

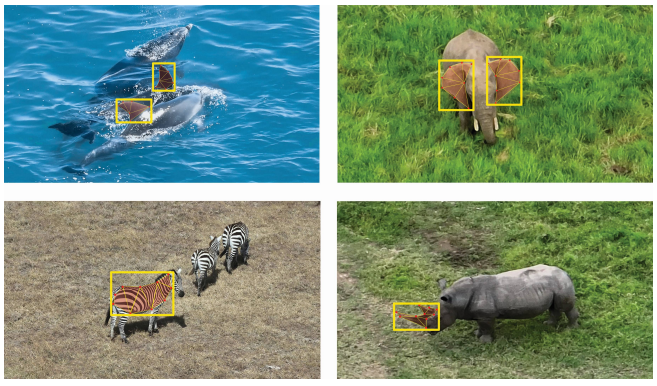


Figure 2: Examples of SIs

http://www.imavs.org/

The lack of uniformity in mission planning for animal ecology studies highlights the need for a general method to assess the quality of data collected regarding the different SIs to be monitored. The autonomous coordination of multiple drones for aerial imagery collection could address these challenges, significantly enhancing both the efficiency and quality of wildlife studies. Both centralised and decentralised computing approaches can be employed to address this challenge. In this study, we focus on a centralised method, which is sufficient for this setting and more feasible to implement for real-world field tests. In the future, we will consider a decentralised approach for surveying multiple herds, as proposed in [25]. These challenges motivate the need for an adaptive, non-invasive approach that can be customised to various drone swarm sizes and SIs.

3 PROBLEM STATEMENT

Our drone swarm deployment problem resembles the *art gallery problem* [30], which focuses on optimising the placement of cameras to cover as much ground area as possible. To address this, we explore solutions from the camera placement literature. Strategies, including the utilisation of a dedicated objective function coupled with bio-inspired algorithms such as Particle Swarm Optimisation (PSO) [31, 32], have been employed to find optimal camera configurations. We base our methodology on this approach and thus need to define an objective function for assessing the monitoring quality of a herd of gregarious animals using the concept of SIs.

Given a set $\mathcal{A}$ of N_a animals distributed across a 2D plane, each animal defined by its position $p_i = (x_i, y_i)$ and its heading θ_i. We define our drone swarm as a set of M agents, represented as $\mathcal{D}$. We aim to find a configuration for the swarm in which the configuration of each drone π_j leads to the maximisation of the value of the objective function. Assuming our drones are stabilised, the camera frustum is determined by the following parameters: drone position in 3D space (u_j), yaw angle (Ψ_j), camera tilt (τ_j), and the horizontal field of view (FoV_j). We define a drone configuration tuple as $s_j = (u_j, \Psi_j, \tau_j, FoV_j)$, and $\mathcal{S} = \{s_1, s_2, \ldots, s_M\}$ as the parameters of the swarm. We also introduce $\overrightarrow{v_j}$, which defines the view direction of the drone. Figure 3 illustrates all the parameters for an animal i and a drone j.

3.1 Design Criteria

We want to construct an objective function, $\Omega(S) \in [0, 1]$, representing the monitoring quality: the closer the value is to 1, the higher the quality. To achieve this, we must define the monitoring objectives and constraints for effective non-intrusive data collection.

Relative angle of the drone view and the animal: The relative angle should be optimised to maximise the visibility of the different SIs on the animal. The drone's view direction $\overrightarrow{v_j}$ should be as close to collinear as possible with the surface normal of these SIs for effective monitoring. Camera traps, which usually capture images from a near-horizontal angle, have shown that individual identification is easier when the images provide a clear, perpendicular view of the relevant characteristics [33].

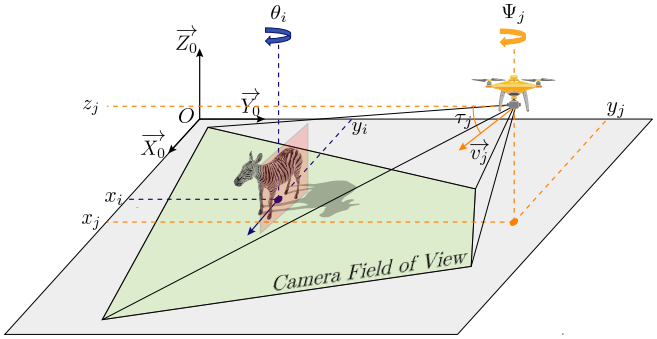


Figure 3: Illustration of parameters for an animal i and a drone j

FoV centring: In the captured images, the animals should be centred in the camera frame. Placing the animals away from the edge of the frame helps prevent distortion and chromatic aberration in the collected data.

Maximising the coverage of the different SIs: The configurations should minimise redundancy by ensuring that different SIs are being monitored by different drones. This means that a solution where multiple drones monitor the same SI provides less information than one where more SIs are covered.

Non-intrusive monitoring: The quality of the collected data should be sufficient in terms of the pixel-to-meter ratio to enable biological studies. However, this must be balanced against the negative impact of wildlife disturbance, as the noise generated by drones poses a potential threat to the well-being of wildlife [13]. Thus, we assume that our drones should not be closer than d_{min} to avoid introducing any disturbances, and not farther than d_{max} to prevent low-quality data collection [10].

3.2 An Analogy with Lambertian Surfaces

To construct our optimisation function, Ω, we drew inspiration from *Lambert's cosine law*. This law states that the intensity reflected by a surface I is proportional to the cosine of the angle θ between the surface normal and the observer's line of sight, as expressed by the equation $I = I_0 \cos\theta$ [14]. By analogy, we relate surface reflectivity to monitoring quality, as this equation, specifically the cosine term, allows us to model the relative angle criterion. For an animal i, our goal is to maximise the overall 'reflectivity' of its SIs, where I_0 in our solution should be extended to take into account the other criteria of Section 3.1.

http://www.imavs.org/

3.3 Monitoring Quality of a Surface of Interest

For each animal i, we associate a set of $N_{L,i}$ SIs, denoted as $\mathcal{T}_i = \{SI_{i,1}, SI_{i,2}, \ldots, SI_{i,N_{L,i}}\}$. Each SI k is associated with a surface centre $O_{i,k}$ and a normal vector $\overrightarrow{n_{i,k}}$. For a SI we want to evaluate the monitoring quality provided by a drone π_j. This assessment involves quantifying the quality of the FoV centring, modelled by the function $\alpha(SI_{i,k}, \pi_j)$, and the pixel-to-meter ratio of the images, modelled by the function $\beta(SI_{i,k}, \pi_j)$. As defined in equations (1) and (2), these functions judge the quality of centring and the distance to the surface through piece-wise continuous functions that yield output values between 0 and 1, indicating how well the quality criteria are met. We define $\mathcal{H} = \{\alpha, \beta\}$ as the set containing these functions, and let N_H denote the number of functions in $\mathcal{H}$.

$$\alpha(SI_{j,k}, \pi_j) = \begin{cases} 1 - \frac{l_{j,k}}{d_{max}} & \text{if } l_{j,k} \leq d_{\max} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$\beta(SI_{i,k}, \pi_j) = \begin{cases} \frac{d_{max} - d_{j,k}}{d_{\max} - d_{\min}} & \text{if } d_{\min} \leq d_{j,k} \leq d_{\max} \\ \frac{1}{d_{j,k}} & \text{if } d_{j,k} < d_{\min} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

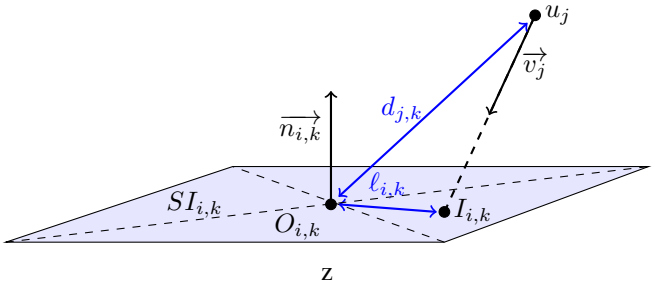


Figure 4: Illustration of parameters for a $SI_{i,k}$ and a drone j

Figure 4 illustrates the parameters of the quality functions, where $l_{j,k}$ represents the distance between $O_{i,k}$ and the intersection point of the ray passing through $\overrightarrow{v_j}$ with $SI_{i,k}$. $d_{j,k}$ represents the distance between the drone and the centre of the surface considered. Thus, we use the Lambertian analogy to construct our monitoring quality function, as presented in equation (3). We calculate the mean of the quality functions to ensure that the output remains between 0 and 1. In this study, we assume that FoV centring and distance have equal importance. Consequently, our model can be expanded if more quality functions are introduced to the set $\mathcal{H}$.

$$\Gamma(SI_{i,k}, \pi_j) = \frac{1}{N_H} \sum_{f \in \mathcal{H}} f(SI_{i,k}, \pi_j) \cdot |\langle \overrightarrow{n_{i,k}}, \overrightarrow{v_j} \rangle| \tag{3}$$

3.4 Generalisation to the Drone Swarm and the Herd

We express our monitoring quality $\Omega(\mathcal{S})$ given a drone swarm configuration $\mathcal{S}$ in equation (4), which we aim to maximise based on the drones' configuration. This Lambertian-inspired approach allows us to define a generic method for assessing the monitoring quality provided by a drone swarm, regardless of the number of animals and their associated surfaces of interest to be monitored. Equation (4) defines the objective function as a product of the monitoring quality $\Delta(S)$ and the penalty $\Lambda(S)$ based on the criteria in Section 3.1.

$$\max_{s_1, s_2, \ldots, s_M} \quad \Omega(\mathcal{S}) = \Delta(\mathcal{S}) \cdot \Lambda(\mathcal{S}) \tag{4}$$

We can compute the *monitoring quality* Δ over the entire herd of animals using equation (5). $\mathcal{D}_{i,k}$ is a subset of the drones in $\mathcal{D}$ representing the drones that can view the k-th surface of interest belonging to the animal i in their camera's field of view. If different drones monitor the same surface, the contribution retained will be that of the drone with the highest individual monitoring quality value, rendering the others redundant.

$$\Delta(\mathcal{S}) = \frac{1}{N_a} \sum_{i=1}^{N_a} \sum_{k=1}^{N_{L,i}} \frac{1}{N_{L,i}} \max_{\pi_j \in \mathcal{D}_{i,k}} \Gamma(SI_{i,k}, \pi_j) \tag{5}$$

Finally, the criteria for developing a *non-intrusive* system is defined as a penalty Λ to our objective function Ω. We apply the penalty Λ to solutions where a drone in the swarm is too close to one of the animals or if a drone is not monitoring any animal, and is presented in equation (6). $\delta_j \in [0, 1]$ applies a possibly non-linear penalty if the drone does not monitor any animal, as seen in equation (7). The subset $\mathcal{A}_j$ of $\mathcal{A}$ represents the animals seen by drone j. The further the drone is from the centre of the herd (represented by the drone's distance to the centroid of the herd, $d_{j,c}$), the greater the penalty. $\phi_{i,j} \in [0, 1]$ applies a linear penalty if drone j is too close to animal i, and can be seen in equation (8).

$$\Lambda(\mathcal{S}, \mathcal{A}) = \prod_{i=1}^{N_a} \prod_{j=1}^{M} \delta_j \phi_{i,j} \tag{6}$$

$$\delta_j = \begin{cases} 1 & \text{if } |\mathcal{A}_j| \neq 0 \\ 1 - \frac{d_{j,c}}{d_{max}} & \text{otherwise} \end{cases} \tag{7}$$

$$\phi_{i,j} = \begin{cases} 1 & \text{if } d_{i,j} > d_{min} \\ \frac{d_{i,j}}{d_{min}} & \text{otherwise} \end{cases} \tag{8}$$

4 EXPERIMENTAL SETUP

Given a swarm of M drones, we aim to find the set of drone configurations $\mathcal{S}$ that maximises our objective function Ω. The optimisation involves $M \times N_S$ parameters. We choose to decrease the degrees of freedom by limiting the number of parameters. We assume that the FoV of the onboard cameras

http://www.imavs.org/

is the same for all drones. For the remainder of this paper, we will use a FoV of $40°$. The drone swarm is divided into two types of roles: one for vertical monitoring and the other for horizontal monitoring. The altitude and camera tilt for these drones will be kept constant. For the vertical monitoring drones, the chosen altitude is d_{min} with a camera tilt of $90°$. For the horizontal monitoring drones, the altitude is $d_{\text{min}}/2$ with a camera tilt of $30°$. Therefore, for each drone j, the parameters to optimise are x_j, y_j, and Ψ_j thus reducing the search space dimension of our optimisation problem.

We assume that the positions and orientations of the animals, as well as the locations of their associated SIs, are known. In this initial approach, the animals are considered static and we do not consider potential environmental obstacles that the drones might encounter in the real world.

4.1 Simulation Platform

We developed a 3D simulator based on the RayLib framework[1] in Python. This open-source tool, designed for creating video games and graphical applications, provides a flexible and highly customisable platform for visualising our studied species, their associated SIs, and their positions on the ground. Furthermore, it allows us to define camera views associated with different drone parameters, as demonstrated in the dedicated YouTube video[2].

In this paper, we concentrate on a case study involving zebras and define three SIs for our analysis. Firstly, we examine the back of the animal, which facilitates behavioural analysis and census tracking. Secondly, we focus on the zebra's left and right longitudinal sides. By monitoring these two surfaces, we can identify individual zebras through side views, which reveal their unique features. To simplify the development, we consider the surfaces that belong to a rectangular prism encompassing the zebra.

4.2 Evaluation Methodology

We implement a centralised controller based on the PSO algorithm and then assess the relevance of our methodology across different herd configurations from real-world data. To build our validation set and draw performance conclusions, we have used Koger's dataset [7], which includes labelled vertical-view pictures of various animal species, including Grevy's zebras. The dataset also provides the drone's telemetry data, allowing us to deduce the animals' planar coordinates. Additionally, we labelled the pictures to retrieve the heading of each animal. The 10 studied herd configurations consist of a mean size of 19.70 animals, with an associated standard deviation of ± 11.02. The standard deviation of the herds' heading varies: in the worst case when the zebras are grazing, it can reach as much as $\pm 140.35°$. Otherwise, when the animals are moving, the standard deviation is smaller, with the smallest being $\pm 19.00°$.

[1] https://github.com/raysan5/raylib
[2] https://www.youtube.com/watch?v=A_eDvyo_bb4



Figure 5: Example of a studied herd configuration

The performance of our approach is assessed using three different drone swarm configurations to evaluate how well it generalises to various setups (# vertical-view drones $\times$ # horizontal-view drones): 1×1, 2×2, and 3×3. These configurations are tested across 10 different animal scenarios. Due to the stochastic nature of the PSO, each experiment is repeated 10 times with different random seeds to assess overall performance, resulting in a total of $10 \times 10 \times 3$ experiments. We retrieve the value of our objective function Ω from these experiments.

5 IMPLEMENTATION OF A CENTRALISED CONTROLLER

We implement a centralised controller to solve the optimisation problem with the objective function Ω and determine the drone configurations $\mathcal{S}$ for deploying them over the zebra herd. This controller calculates a configuration for each drone based on the zebras' spatial distribution and their respective SIs. As detailed in Section 2, we use PSO to determine the (x_j, y_j) coordinates and the yaw Ψ_j for each drone j in the swarm.

5.1 Particle Swarm Optimisation: Bio-Inspired Algorithm

The PSO algorithm [34] is a biologically inspired population-based stochastic optimisation algorithm that explores a search space to find the best solution. In PSO, each particle represents a candidate solution and adjusts its position based on both its own experience and that of the group. The goal is for all particles to converge to an optimal solution by balancing exploration with refinement. Like most optimisation algorithms, PSO has hyperparameters that must be tuned to maximise performance. For PSO, these hyperparameters are defined as follows [34]:

Swarm Size (n): The number of particles exploring the search space. This value balances computational complexity (larger swarms) with solution quality (smaller swarms may lead to premature convergence on local minima).

Inertia Weight (w): Controls the evolution of particle speed at each iteration. A high inertia weight encourages explo-

ration, while a low value promotes the exploitation of local minima.

Cognitive Coefficient (c_1): Reflects a particle's tendency to return to its personal best position.

Social Coefficient (c_2): Represents the influence of the swarm's best-known position on each particle, promoting collaboration.

5.2 Hyperparameter Tuning

To tune the hyperparameters, we follow the recommendations suggested by Robison et al.[34]. We use a swarm size of 100 particles and set the PSO algorithm to 100 iterations. For the remaining parameters w, c_1, and c_2—we employ Bayesian optimisation [35]. In each Bayesian optimisation iteration, we aim to maximise Ω using two vertical and two horizontal drones across 16 herd configurations, ranging from compact to sparse herds. To carry out the tuning, we model the spatial distribution of zebras with a simplified approach, assuming normal distributions for planar positions and headings to ensure the PSO's general applicability. For the 16 synthetic herds, we define a fixed herd size of 30 zebras, in line with the literature [36], and use normal distributions with standard deviations of 10, 20, 30, and 40 metres for planar coordinates, and 0, $\frac{\pi}{4}$, $\frac{\pi}{2}$, and π for headings. The optimisation process, set for 500 iterations, converged by the 21st iteration, resulting in optimal values of $w = 0.53$, $c_1 = 2.27$, and $c_2 = 0.99$.

6 RESULTS AND DISCUSSION

Using the hyperparameters identified in Section 5.2, we validate our approach on the 10 different real-world herd configurations described in Section 4.2. For all the spatial distributions, we observe the same trend: having more drones resulted in a higher objective value than having fewer drones, as the drones could cover more SIs. Moreover, the PSO-based approach is stable in solving the optimisation problem regardless of the number of drones in the swarm. The highest observed standard deviation of the objective function Ω was 0.07. Table 1 details our algorithm's performance for a representative herd configuration consisting of 48 animals. The $\overline{\Omega}$ values presented are the mean values from the 10 experiments conducted.

Swarm Configuration	1x1	2x2	3x3
Objective Function Value $\overline{\Omega}$	0.43	0.66	0.76
Ω Standard Deviation	0.00	0.02	0.01
Top SIs Monitored (%)	83.33	100.00	100.00
Left SIs Monitored (%)	24.58	89.58	98.75
Right SIs Monitored (%)	68.13	91.46	99.17

Table 1: Monitoring results for a herd of 48 animals

A single horizontal drone cannot effectively monitor both sides of an animal simultaneously. As illustrated in the 1×1 column of Table 1, this limitation results in lower monitoring quality for the horizontally observed SIs. Moreover, a vertical drone might struggle to cover an entire herd, affecting the overall monitoring quality. When adding more vertical and horizontal monitoring drones, the generated drone configurations achieve significantly better results, as seen in Table 1.

Moreover, as illustrated in Figure 6, we can qualitatively assess whether the swarm configuration is coherent. In Figure 6a, the number of drones performing the vertical and horizontal views is not sufficient to cover the entire herd; as a result, the drones are monitoring the part of the herd with the highest density, thus maximising the number of SIs covered. In Figure 6b, the vertical drones monitor the two clusters of animals, while the horizontal drones provide complementary views by facing each other. In Figure 6c, the swarm configuration is more complex and allows for an increased monitoring quality value.

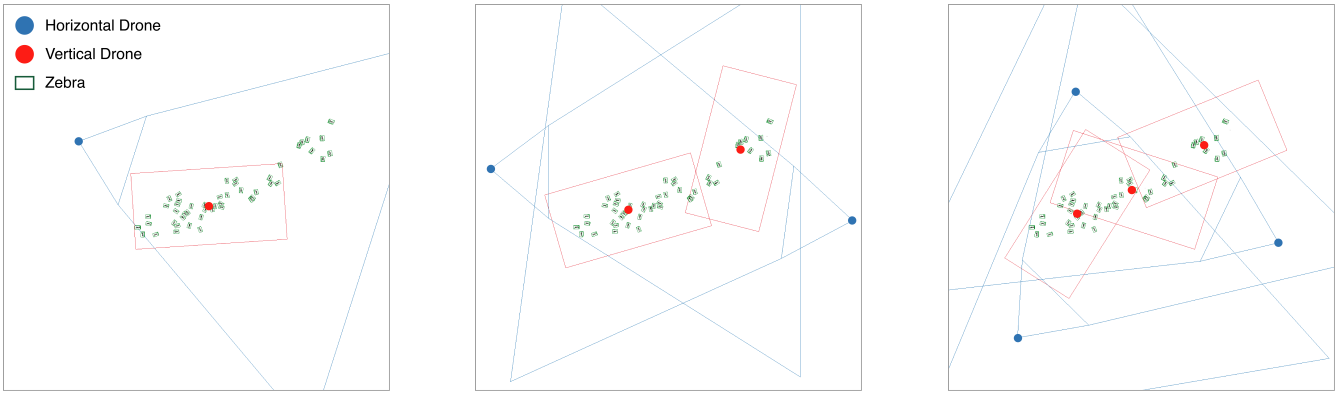
Our approach is based on strong assumptions: the animals are static, and the positions of their SIs are known. To apply this approach in the real world with moving animals, we need to investigate the performance of the PSO in terms of convergence time. If the computation speed is sufficiently high, it might be feasible to run the PSO at a frequency high enough to update the positions of the drones while the animals are moving.

Further work is required to integrate machine vision for detecting and localising the animals. In our centralised approach for real-world deployment, we envision locating the animals using either a coverage algorithm or a drone flying high above the herd [37]. Subsequently, This drone would retrieve the animals' positions and orientations, as well as estimate their SIs. This data would be then processed by a ground station using the PSO algorithm, which updates the drones' positions based on its solution.

7 CONCLUSION

In this paper, we described and formalised a general approach for performing gregarious animal monitoring with a drone swarm based on the concept of surfaces of interest (SI) with Lambertian properties. The results show that our PSO centralised controller can find a drone configuration that maximises the quality of monitoring the herd, for different swarm size. In addition, we evaluated the performance of the PSO by analysing the drone configurations obtained using a real-world dataset of zebra distributions. Our algorithm was able to generate satisfactory configurations that met the criteria outlined in Section 3.1, using as few as two drones and up to six drones. As part of the WildDrone Project [38], we plan to test our approach in Kenya in January 2025. To achieve this goal, further work is needed to quantify the performance of our method in terms of computing speed and to enable animal localisation through machine vision.

http://www.imavs.org/



(a) 1x1 (b) 2x2 (c) 3x3

Figure 6: Drone configurations for 1x1, 2x2, and 3x3 vertical and horizontal monitoring setups

ACKNOWLEDGEMENTS

This work is supported by the WildDrone MSCA Doctoral Network funded by EU Horizon Europe under grant agreement no. 101071224, the Innovation Fund Denmark for the project DIREC (9142-00001B), the Imageomics Institute, funded by the US National Science Foundation (2118240), and the ICICLE Institute funded by the National Science Foundation (OAC-2112606).

REFERENCES

[1] Gerardo Ceballos, *et al.* Biological annihilation via the ongoing sixth mass extinction signaled by vertebrate population losses and declines. *Proceedings of the National Academy of Sciences*, 114(30):E6089–E6096, 2017.

[2] Marie RG Attard, *et al.* Review of satellite remote sensing and unoccupied aircraft systems for counting wildlife on land. *Remote Sensing*, 16(4):627, 2024.

[3] Edouard G. A. Rolland, *et al.* Autonomous UAV volcanic plume sampling based on machine vision and path planning. In *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1064–1071. IEEE, 2024.

[4] Evangeline Corcoran, *et al.* Automated detection of wildlife using drones: Synthesis, opportunities and constraints. *Methods in Ecology and Evolution*, 12(6):1103–1114, 2021.

[5] Kasey P Ryan, *et al.* Use of drones for the creation and development of a photographic identification catalogue for an endangered whale population. *Arctic Science*, 8(4):1191–1201, 2022.

[6] Lukas Schad. Opportunities and risks in the use of drones for studying animal behaviour. *Methods in Ecology and Evolution*, 14(8):1864–1872, 2023.

[7] Benjamin Koger, *et al.* Quantifying the movement, behaviour and environmental context of group-living animals using drones and computer vision. *Journal of Animal Ecology*, 92(7):1357–1371, 2023.

[8] Sota Inoue, *et al.* Spatial positioning of individuals in a group of feral horses: A case study using drone technology. *Mammal Research*, 64:249–259, 2019.

[9] Maksim Kholiavchenko, *et al.* KABR: In-Situ Dataset for Kenyan Animal Behavior Recognition from Drone Videos. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops*, pages 31–40. IEEE, 2024.

[10] Sarah Sofía Landeo-Yauri, *et al.* Using small drones to photo-identify antillean manatees: A novel method for monitoring an endangered marine mammal in the Caribbean sea. *Endangered Species Research*, 41:79–90, 2020.

[11] Mayank Lahiri, *et al.* Biometric animal databases from field photographs: identification of individual zebra in the wild. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, ICMR '11, pages 1–8. Association for Computing Machinery, 2011.

[12] Luca Bergamini, *et al.* Multi-views embedding for cattle re-identification. In *2018 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, pages 184–191. IEEE, 2018.

[13] Saadia Afridi, *et al.* Unveiling the impact of drone noise on wildlife: A crucial research imperative. In *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1409–1416. IEEE, 2024.

[14] Michael Oren. Generalization of Lambert's reflectance model. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*,

http://www.imavs.org/

SIGGRAPH '94, page 239–246. Association for Computing Machinery, 1994.

[15] Marc Besson, *et al.* Towards the fully automated monitoring of ecological communities. *Ecology Letters*, 25(12):2753–2775, 2022.

[16] Lacey F. Hughey, *et al.* Challenges and solutions for studying collective animal behaviour in the wild. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 373(1746):20170005, 2018.

[17] Devis Tuia, *et al.* Perspectives in machine learning for wildlife conservation. *Nature Communications*, 13(1):1–15, 2022.

[18] Andrew J King, *et al.* Biologically inspired herding of animal groups by robots. *Methods in Ecology and Evolution*, 14(2):478–486, 2023.

[19] Jeanne Altmann. Observational study of behavior: Sampling methods. *Behaviour*, 49(3–4):227–266, 1974.

[20] Vandita Shukla, *et al.* Towards estimation of 3d poses and shapes of animals from oblique drone imagery. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLVIII-2–2024:379–386, 2024.

[21] Kevin Charles Bierlich, *et al.* Automated body length and body condition measurements of whales from drone videos for rapid assessment of population health. *Marine Mammal Science*, page e13137.

[22] Eduard Degollada, *et al.* A novel technique for photo-identification of the fin whale, Balaenoptera physalus, as determined by drone aerial images. *Drones*, 7(3):220, 2023.

[23] William Andrew, *et al.* Aerial animal biometrics: Individual Friesian cattle recovery and visual identification via an autonomous UAV with onboard deep inference. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 237–243. IEEE, 2019.

[24] Wei Luo, *et al.* An efficient visual servo tracker for herd monitoring by UAV. *Scientific Reports*, 14(1):10463, 2024.

[25] Denys Gruschak, *et al.* Decentralized multi-drone coordination for wildlife video acquisition. In *5th IEEE International Conference on Autonomic Computing and Self-Organizing Systems*. IEEE, 2024. In press.

[26] Mohammed A Alanezi, *et al.* Livestock management with unmanned aerial vehicles: A review. *IEEE Access*, 10:45001–45028, 2022.

[27] Xiaohui Li. Reactive deployment of autonomous drones for livestock monitoring based on density-based clustering. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2421–2426. IEEE, 2019.

[28] Ivana Strumberger, *et al.* Static drone placement by elephant herding optimization algorithm. In *2017 25th Telecommunication Forum (Telfor)*, pages 1–4. IEEE, 2017.

[29] Chen Zhang, *et al.* 3d deployment of multiple UAV-mounted base stations for UAV communications. *IEEE Transactions on Communications*, 69(4):2473–2488, 2021.

[30] Frank Hoffmann. On the rectilinear art gallery problem. In Michael S. Paterson, editor, *Automata, Languages and Programming*, pages 717–728. Springer, 1990.

[31] Yacine Morsly, *et al.* Particle swarm optimization inspired probability algorithm for optimal camera network placement. *IEEE Sensors Journal*, 12(5):1402–1412, 2011.

[32] Xiaohui Wang, *et al.* Solving optimal camera placement problems in IoT using LH-RPSO. *IEEE Access*, 8:40881–40891, 2020.

[33] Tinao Petso, *et al.* Review on methods used for wildlife species and individual identification. *European Journal of Wildlife Research*, 68(1):3, 2022.

[34] J. Robinson. Particle swarm optimization in electromagnetics. *IEEE Transactions on Antennas and Propagation*, 52(2):397–407, 2004.

[35] Jia Wu, *et al.* Hyperparameter optimization for machine learning models based on Bayesian optimization. *Journal of Electronic Science and Technology*, 17(1):26–40, 2019.

[36] Alfred Stein. *Spatial Marked Point Patterns for Herd Dispersion in a Savanna Wildlife Herbivore Community in Kenya*, pages 261–273. Springer, 2006.

[37] Kasper A. R. Grøntved, *et al.* Decentralized multi-UAV trajectory task allocation in search and rescue applications. In *2023 21st International Conference on Advanced Robotics (ICAR)*, pages 35–41. IEEE, 2023.

[38] Edouard Rolland, *et al.* Advancing wildlife monitoring in gregarious species with drone swarms. In *Distributed Computing and Artificial Intelligence, 21st International Conference*. Springer, 2024. In press.

http://www.imavs.org/

WildBridge: Conservation Software for Animal Localisation using Commercial Drones

K. Meier*, A. Richards, M. Watson, G. Maalouf, C. Johnson, D. Hine and T. Richardson
University of Bristol, Queens Road, Bristol, BS8 1QU, United Kingdom

http://www.imavs.org/

ABSTRACT

This paper assesses the usability and accuracy of a commercially off-the-shelf (COTS) platform for precise terrestrial animal localization. To achieve localization, this paper uses a single-shot coordinate transform method. The platform selected in this work is the DJI Mavic 3E quadcopter, since it is a popular and widely adopted platform, in particular among the wildlife conservation community. To enable the use of the DJI Mavic 3E, a software interface was developed to access telemetry and camera feed in realtime. The method was tested during two field campaigns in Kenya and Wales. These field campaigns permitted the discovery of localization error sources and their possible mitigation, in particular through adequate mission planning. The preliminary maximum localization error was computed to be 19.2m laterally and 14.4m longitudinally for an animal positioned 385m from the UAV flying at 120m above ground.

1 INTRODUCTION

1.1 Overview

Recent developments in Computer Vision (CV) and Unoccupied Aircraft System (UAS) have offered the potential for providing a step-change in animal conservation through automated monitoring of wildlife from the air – yet despite many successful deployments, the true potential of the concept has yet to be realised [1].

This paper will provide a stepping stone on the front of animal detection and localization. More specifically, this paper has three technical aims: (i) interface with a commercial platform (DJI Mavic 3E) from a local computer to provide real-time detection and localisation of large mammals, (ii) evaluate the localisation accuracy of the observed animals and identify the error sources, (iii) provide recommendation for the deployment of this system in the field.

WildBridge is the name of the system that brings these three goals together to create an integrated system for real-time localization of animals using Commercial Off The Shelf (COTS) drones. The WildBridge user interface can be seen in Figure 1.

*Email address: kilian.meier@bristol.ac.uk

1.2 Background

It is estimated that one-fifth of the world's vertebrate species are currently threatened with extinction [2], consequently, the need for more effective means of surveying wildlife populations has never been so important. Wildlife is traditionally monitored primarily either by foot, car, ship, crewed aircraft, or more recently using camera traps. However, these methods are often expensive, time-consuming, and logistically difficult in remote areas, making the monitoring and assessment of biodiversity one of the biggest challenges in modern conservation. Effective wildlife monitoring and adaptive management critically depend on regular surveys of wildlife abundance [3]. Aerial monitoring has the potential to provide frequent and systematic surveys.

There are three primary means by which aerial imagery can be gathered for conservation purposes: crewed aircraft, satellites and UAS/drones.

For large African landscapes, manual aerial surveys with light aircraft are still considered the best method of counting large mammals [4], [5]. However, high costs and logistical constraints mean that light aircraft surveys either do not take place at all, or if they do, the time between successive surveys is so long that catastrophic declines in populations can occur [6].

Commercially available high-resolution imagery from satellite sensors including Quickbird, GeoEye-1 and World View-3 and 4 have been successful in estimating abundances of large mammals in savannahs [7], [8]. However, obstruction of view caused by vegetation in complex wooded savannah environments limits their application and the lack of very high-resolution imagery prevents individual animal identification [8].

Unlike satellite imagery, aerial surveys using UAS/drones can provide an oblique view in complex habitats providing superior images for wildlife detection and the identification of individual animals [9]. UAS can be deployed quickly, fly autonomously, and collect high-resolution aerial imagery, making them a key candidate for detecting and counting wildlife [10]. In the last 20 years, UAS technologies have advanced substantially and the subsequent decreases in cost have made them more readily accessible in the field of conservation. Consequently, several key areas have been highlighted within conservation where UAS might provide great benefit, e.g. the real-time mapping of land cover, monitoring of illegal deforestation, detection of poaching activities, and wildlife surveying [11], [12], [13]. More specifically,

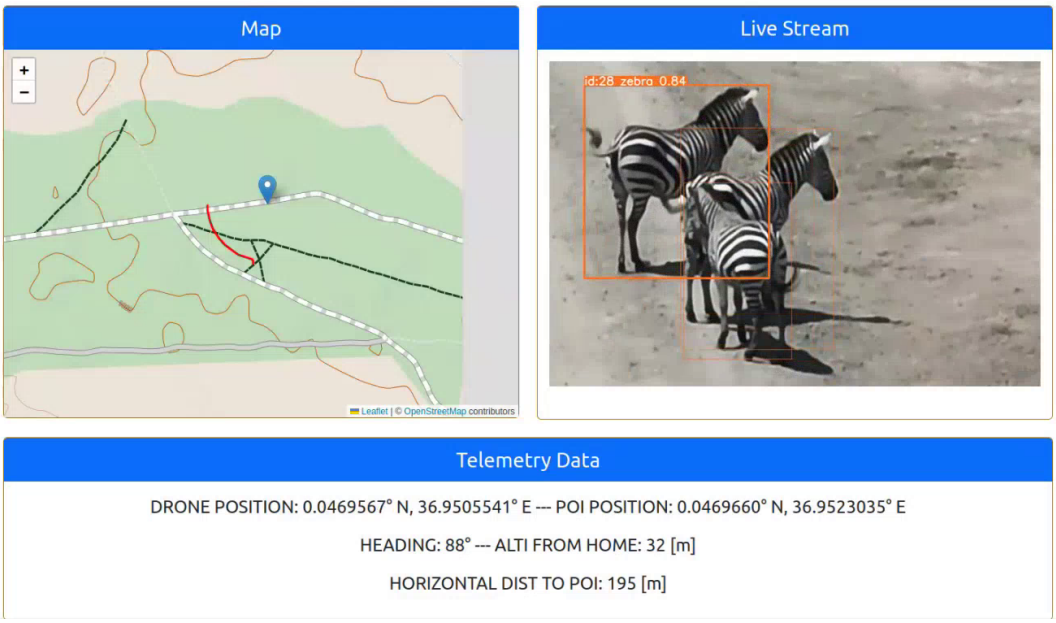


Figure 1: WildBridge web-based graphical user interface. The left tile shows the localization results on a map: the red line is the drone trajectory (the drone is at the top left of the curve) and the blue pin represents the estimated location of the animal. The right tile shows the live camera feed and highlights the localized animal. The bottom tile displays selected telemetry. This snapshot was taken during a flight in Ol Pejeta Conservancy, Kenya.

http://www.imavs.org/

ecologists have successfully used UAS to survey a variety of taxa, including several bird species (Canada geese, snow geese, and white ibis) [14]; great ape nests [15]; salmon [16]; and manatees [16]. A variety of sensors have also been utilized with good effect including thermal [17] and audio recorders [18]. Recent developments in deep learning image recognition can now automatically count animals from aerial images in open and homogenous landscapes [19], [20].

1.3 Problem Formulation

The task at hand is to automatically and in real-time determine the geographic location of the POI, given the knowledge of the drone's geographic location, the camera orientation, the camera parameters and the location of a POI in the image. To access the data needed for the localisation, a system must be built to have real-time access to a drone's telemetry and camera feed. In order to make this system easy to use, robust and accessible, three main design choices have been made. First, WildBridge has been developed for the DJI Mavic 3E quadcopter (see Section 2.2), which is a widely used platform in conservation due to its price, ease of use and robustness [21], [22]. Second, the software stack has been mainly written in Python, which is a popular language due to its ease of use and allows WildBrige to be easily upgraded and customized according to the user's need.

The rest of the paper will be organised as follows. Section 1.4 explores the state of the art, further assessing the need for a tool like WildBridge, as well as listing the different tools and methods already in use by conservationists. Section 2 provides an overview of the different components of WildBridge and how they interact. This section also presents how WildBridge can be adapted to different platforms and tailored for different needs. Section 3 describes how the localisation is performed. Section 4 presents preliminary results based on data collected by WildBridge and discusses the nature and magnitude of measurement errors. Finally, Section 5 concludes and presents further development directions of WildBridge.

1.4 State of the Art

Several research groups have tried to address the question of localizing animals using UAS. In particular, several conservation groups have used mapping techniques or orthographic photos to count, locate and measure animals in optical images: such as various crocodylians [21], European brown hare (*Lepus europaeus*) [23] or Black-Necked Swan (*Cygnus melancoryphus*) [24]; and thermal images: northern bobwhite quail (*Colinus virginianus*) [17]. The benefit of using nadir imagery is the simplification of the problem's geometry and ensures a well-known Ground Sample Distance (GSD), thus enabling size measurements [21]. It also leverages well-established mapping tools such as Pix4D (Prilly, Switzerland, pix4d.com). However, generating a map is computationally intensive and does not allow real-time localization and mapping. Moreover, the requirement for nadir images and a specific overlap constrain flight operations and

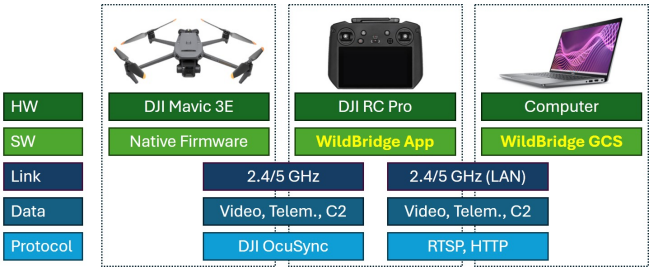


Figure 2: WildBridge system block diagram. store.dji.com, etrix.com.sg

does not give the user the flexibility to react to unexpected changes during the survey.

A different problem which has, however, a related solution is the problem of UAS localization in GNSS-denied environments. There are several approaches to solve this problem, but one of them is triangulating the UAS position from known reference points [25] and [26]. The geometry of the problem is the same, but the unknown is the UAS position instead of the animal position (reference point).

It is also worth mentioning that using image-based techniques is not the only viable approach to animal localization. Radio-based approaches have also proven to be effective [27] and have led to companies offering it as a service, such as Wildlife Drones (Acton, Australia, wildlifedrones.net). Radio location has the benefit of not relying on any resource-intensive image processing pipeline, of being able to identify specific individuals, and of permitting reliable localization in occluded and dense environments. It comes however at the cost of having to tag the animals to be tracked, which might be difficult or even impossible. DJI has also released a laser-based approach, enabling the localisation of POIs seen by the drone (DJI PinPoints). To achieve this, a laser range finder is used to measure the distance between the drone and the POI. This distance is then used to localise the POI based on the drone's position and orientation.

Finally, authors in [28] investigated the use of UAS to locate invasive trees (*Miconia calvescens*) using a similar approach to the one presented in this work. However, in [28], the localization is performed in post-processing.

Note that most papers cited in the section use DJI drones, which suggests their adequacy for wildlife conservation applications.

2 EXPERIMENTAL SETUP

2.1 Overview

Figure 2 shows an overview of WildBridge and the following sections describe each component of the system.

2.2 Drone platform

The UAV selected for this work is the DJI Mavic 3 Enterprise (Figure 2). A summary of its specifications can be found in Table 1. The platform was selected due to its wide use and its proven record in remote and harsh environments. In addition, it features high-quality cameras with a zoom capacity of up to x56, which is ideal for animal surveys while avoiding any disturbances on the animals. While this platform has been selected for its high performance, WildBridge should be compatible with all DJI products supporting Mobile SDK V5[1], i.e. Mavic 3, Mini 3 and Matrice 300 series (see Section 2.3 for more details about the SDK). However, WildBridge has not been tested on any of those platforms yet.

Aircraft	Weight	915 g
	Diagonal	380.1 mm
	Max hover time	38 min
Wide cam.	Sensor	4/3 CMOS, 20 MP
	Lens FOV	84°
	Lens format equi.	24mm
	Lens aperture	f/2.8-f/11
	Video resolution	FHD or 4K at 30fps
Tele cam.	Sensor	1/2-inch CMOS, 12 MP
	Lens FOV	15°
	Lens format equi.	162mm
	Lens aperture	f/4.4
	Video resolution	FHD or 4K at 30fps
	Digital Zoom	8x (56x hybrid zoom)

Table 1: DJI Mavic 3E specifications. Summarised from https://enterprise.dji.com/mavic-3-enterprise/specs.

2.3 Remote Controller (RC)

The RC selected for this work is the DJI RC Pro (Figure 2. This is the default controller delivered with the DJI Mavic 3E. Importantly, the RC runs on Android and supports DJI's Mobile SDK V5 (MSDK). This Software Development Kit (SDK) provides an interface to the drone and thus enables the WildBridge app to access the drone's telemetry and video streams, and send control commands back to the drone. The RC communicates with the drone over a 2.4 GHz and a 5 GHz radio link and with the GCS laptop via a Local Area Network (LAN).

2.4 WildBridge app

The WildBridge app is the actual "bridge" of the project. It enables any device on the LAN to receive telemetry data and to send control commands from and to the drone. This is achieved by implementing an HTTP server answering to telemetry and control command requests. The WildBridge app also runs an RTSP server broadcasting the drone's camera feed. The app is written in Java and Kotlin and has been adapted from the MSDK's example application[2]. The telemetry fields accessed in this work over the MSDK are KeyAircraftLocation3D, KeyAircraftAttitude, KeyGim-

[1] https://developer.dji.com/mobile-sdk/
[2] Mobile-SDK-Android-V5

http://www.imavs.org/

balAttitude and KeyCameraZoomRatios[3]. Note, that this implies that the drone cannot be flown using the native Pilot 2 app and the WildBridge app must be used instead for piloting.

2.5 Laptop

In this work, a Dell Inc.Latitude 5440 operating on Ubuntu 22.04.4 LTS was used and was able to perform around 4 to 5 localizations per second. However, most modern computers can be used as long as they can access the LAN and can run Python scripts. Given that the computer is running an object detection algorithm it is recommended the computer has reasonable image processing capabilities. This machine also hosts the web-based user interface, displaying the localization results (Figure 1).

3 METHODOLOGY

3.1 Notations and reference frames

A vector $\mathbf{r}$ in a frame f is noted as $\mathbf{r^f}$. A rotation matrix $\mathbf{C}$ transforming a vector from frame a to b is noted as $\mathbf{C_a^b}$, i.e. $\mathbf{r^b} = \mathbf{C_a^b r^a}$.

There are three reference frames to consider: the image frame (i), the camera frame (c) and the world frame (w). The image frame is a two-dimensional frame, spanning the camera sensor plane. Its origin is in the top left corner of the image. The camera frame has its origin at the focal point of the camera and is oriented such that its y- and z-axis are parallel to the x- and y-axis of the image frame, respectively. The camera frame's x-axis is perpendicular to the sensor plane and crosses it in its centre (see Figure 3). The distance between the origin of the camera frame and sensor plane and the focal length. Finally, the world frame's x-, y- and z-axis are aligned with the North, East and Down cardinal directions.

3.2 Localisation

The localisation problem can formulated as finding the relationship between a POI detected in the image and a POI in the world. Let's call the former the virtual POI (vPOI) and the latter the world POI (wPOI).

Assuming a vPOI has been found in the image frame, locating it in the world can be achieved by tracing a ray originating from the camera focal point (origin of the camera frame) and passing through the vPOI. The intersection between this ray and the ground is the location of the wPOI (see Figure 3). Hence, this section needs to derive a parameterization of this ray in the world frame. Thus, let's assume a known vPOI in the image frame such that:

$$\mathbf{r^i_{vPOI}} = \begin{bmatrix} x^i_{vPOI} \\ y^i_{vPOI} \end{bmatrix} \tag{1}$$

where x^i_{vPOI} and y^i_{vPOI} are expressed in meters. However, the position of the detected POI in the image is measured in pixels. The conversion can be achieved by knowing the physical size of a pixel in the camera sensor. For the zoom lens of the DJI Mavic 3E used in this work, the pixels have a length $s = 1.6\ [\mu m]$[4]. Thus:

$$\mathbf{r^i_{vPOI}} = \begin{bmatrix} x^i_{vPOI} \\ y^i_{vPOI} \end{bmatrix} = s \cdot \mathbf{r^i_{vPOI,pix}} \tag{2}$$

where $\mathbf{r^i_{vPOI,pix}}$ is the position of the vPOI in the image frame expressed in pixels.

Then the position of the vPOI in the camera frame can be expressed as:

$$\mathbf{r^c_{vPOI}} = \begin{bmatrix} f \\ x^i_{vPOI} - W/2 \\ y^i_{vPOI} - H/2 \end{bmatrix} \tag{3}$$

where W and H are the total image width and height respectively and f the camera's focal length. Hence a parametric line $\mathbf{d}$ originating from the camera origin and passing through the vPOI can be written as:

$$\mathbf{d}(\lambda)^\mathbf{c} = \lambda \mathbf{r^c_{vPOI}} \tag{4}$$

Where λ is the parameter. Now let's express this line in the world frame:

$$\mathbf{d}(\lambda)^\mathbf{w} = \lambda \mathbf{C_c^w r^c_{vPOI}} + \mathbf{r_c^w} \tag{5}$$

where $\mathbf{r_c^w} = [n_c\ e_c\ d_c]^T$ is the known translation of the camera frame with respect to the world frame and $\mathbf{C_c^w}$ is the rotation matrix encoding the relative orientation of the world

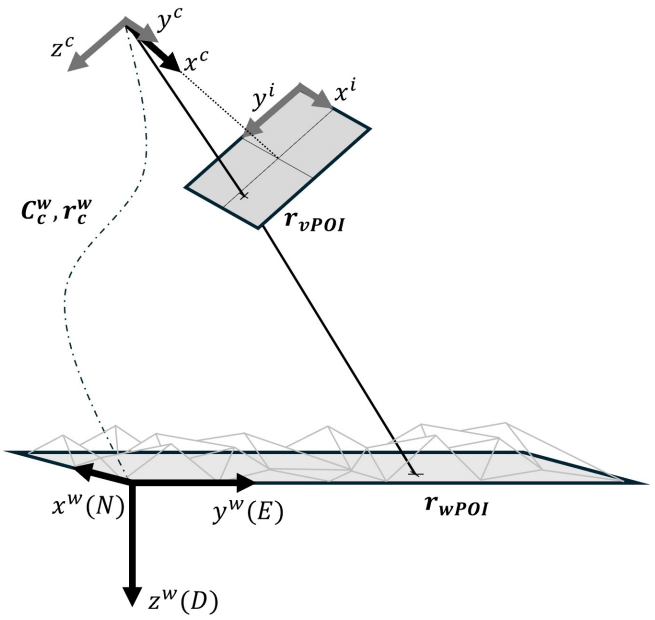


Figure 3: Reference Frames.

[3]DJI MSDK Documentation

[4]Calculated from the surface each pixel covers based on the DJI Mavic 3E specifications.

http://www.imavs.org/

frame with respect to the camera frame. This rotation can be expressed as an intrinsic "zy" rotation[5]:

$$\mathbf{C_c^w} = \mathbf{C_z}(\psi)\mathbf{C_y}(\theta) = \begin{bmatrix} c_\psi c_\theta & -s_\psi & c_\psi s_\theta \\ s_\psi c_\theta & c_\psi & s_\psi s_\theta \\ -s_\theta & 0 & c_\theta \end{bmatrix} \quad (6)$$

where ψ is the camera heading, θ is the camera pitch and s and c represent the sine and cosine functions of the subscripted angle. Finally, the intersection between $\mathbf{d}(\lambda)^\mathbf{w}$ and the ground can be expressed as:

$$\mathbf{d}(\lambda)^\mathbf{w} = \mathbf{r_{wPOI}^w} = \begin{bmatrix} n_{wPOI} \\ e_{wPOI} \\ d_{wPOI} \end{bmatrix} \quad (7)$$

where n_{wPOI} and e_{wPOI} are the northing and easting of the wPOI, which are unknown, and d_{wPOI} the elevation of the wPOI. The elevation could be known using a digital elevation model of the area of interest and then finding the intersection of the traced ray and the elevation map. However, this is outside of the scope of this paper. Hence, it is assumed the terrain is flat and since the POIs are animals evolving on the ground $d_{wPOI} = 0$. This is a strong assumption and a major source of error and will be address in future work. Thus, assuming $d_{wPOI} = 0$, (7) is a system of three equations with three unknowns: λ, n_{wPOI} and e_{wPOI}. Solving for λ leads to:

$$\lambda = \frac{d_c}{c_\theta(y^i_{vPOI} - H/2) - s_\theta f} \quad (8)$$

Explicit solutions for n_{wPOI} and e_{wPOI} can be expressed using (5) and (7). This system is solvable as long as:

$$c_\theta(y^i_{vPOI} - H/2) - s_\theta f \neq 0 \quad (9)$$

i.e. as long as the ray is not horizontal. The system has 11 parameters: x^i_{vPOI}, y^i_{vPOI}, s, f, W, H, n_c, e_c, d_c, ψ and θ.

4 RESULTS AND DISCUSSION

4.1 Ol Pejeta (Kenya) test campaign - Proof of concept

WildBridge has been tested during two different test campaigns with different aims and objectives. The first campaign took place in the Ol Pejeta Conservancy in Kenya and aimed at testing WildBridge in an environment representative of what conservationists may experience during their fieldwork. It was also an opportunity to present this work to the rangers and researchers of Ol Pejeta and discuss potential further developments of WildBridge.

Four flights were performed at various locations in the park, mainly observing zebras. The flights were conducted within visual line of sight (VLOS) since the WildBridge app does not yet implement some critical safety features to enable Beyond Visual Line of Sight (BVLOS) flights. To ensure the regulatory compliance of those flights, a Specific Operational Risk Assessment (SORA) [29] was submitted to both the Kenya Civil Aviation Authority (KCAA) [6] and the Kenyan military, since Ol Pejeta is a military controller airspace.

A snapshot of the WildBridge interface during one of the flights can be seen in Figure 1. Some flights were also live-streamed using Starlink to Ol Pejeta's control room, thus giving live information about animals in the park to the rangers. Note, however, that the accuracy of the localization was not assessed here due to the challenges related to the acquisition of a ground truth.

The main takeaways from this campaign are:

- The system was successfully used in a remote environment with real animal observations.
- The system awoke the interest of Ol Pejeta's rangers and has the potential to be integrated with Earthrange[7], Ol Pejeta's current monitoring interface.
- The WildBridge app, while being useable, should improve its user interface to enable safer piloting of the drone.
- The regulatory approval was more complex and time-consuming than expected due to the involvement of multiple stakeholders. Future missions should allocate sufficient contingency time for potential delays.

4.2 Llanbedr (Wales) test campaign - Characterisation

The second test campaign took place at Llanbedr Airfield in Wales and aimed at evaluating and characterising the localisation error of WildBrige. To achieve this, several flights were performed looking at a stationary POI on the ground, placed at a known location. The flights were flown manually and such that the POI was kept in the frame. All flights were achieved by controlling only two of the five control inputs available on the DJI RC Pro and they can be grouped into four categories: straight horizontal flight away or towards the POI (vertical left stick and gimbal pitch), straight ascent or descent (vertical right stick and gimbal pitch), diagonal flight away or towards the POI (vertical right and left stick), and orbit around the POI (horizontal left and right stick). In addition to those four flight types, flights were also performed only using one of the five available control inputs, but they are limited in range since they cannot control the POI to stay in the frame. These flights will be used in two different ways, first to understand and compensate for the limitations of the localisation method, second to make a first evaluation of the accuracy this system can achieve. Two sample flights can be seen in Figure 8 and 9.

[5]The camera is controlled such that its roll is always zero, hence the absence of an "x" rotation.

[6]https://kcaa.or.ke/

[7]EarthRanger website

http://www.imavs.org/

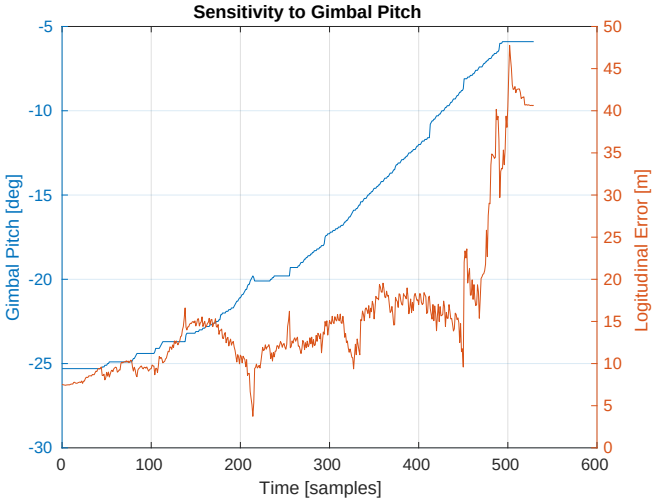


Figure 4: Correlation between pitch angle and longitudinal error during a vertically descending flight.

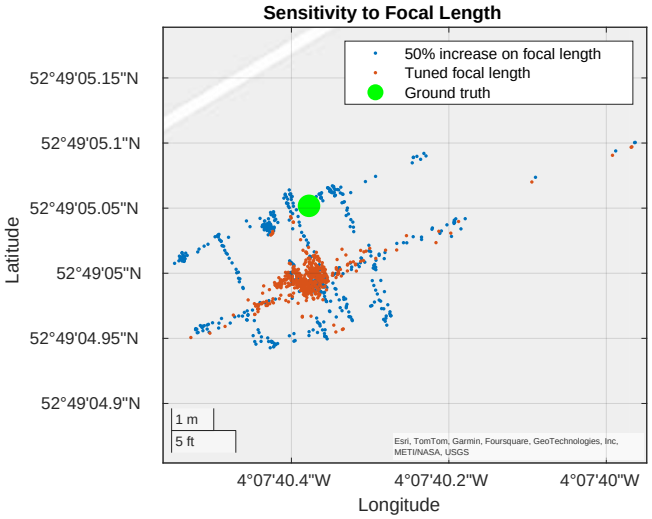


Figure 5: Impact of focal length tuning on localisation estimate during a stationary flight featuring yawing and gimbal pitching movements.

4.3 *Error definition*

The performance metric chosen in this work is lateral and longitudinal error with respect to the azimuth between the drone and the ground truth. In other words, the longitudinal error is the projection of the error vector on the horizontal axis connecting the ground truth point and the drone's position on the ground. The lateral error is the projection on an axis perpendicular to the longitudinal axis.

4.4 *Sensitivity to gimbal pitch error*

As described in Section 3.2, (7) is only solvable when the ray is not horizontal. But it is also worth noting, that for rays close to horizontal, the denominator in (8) is close to zero, making λ very sensitive to small errors in gimbal angle or vPOI detection and thus leading to large errors in the position estimation of the wPOI. This can be illustrated by looking at a vertically descending flight as shown in Figure 4. During the descent from approximately 120m to 20m above ground, the gimbal is operated such that the POI always stays in the field of view. As expected, the longitudinal error increases with decreasing gimbal angle.

4.5 *Sensitivity to focal length*

Unless the vPOI is in the centre of the image, the localisation estimate is dependent on the focal length (and more generally on the camera model used). DJI's MSDK enables access to several measurements of focal length, but after inspection of the data, it is unclear how any of the available values relate to the true focal length of the camera. Hence, in this work, the focal length used is the telemetry field named *KeyCameraZoomRatios*[8] scaled by a manually tuned parameter such that the localisation error is minimized. To illustrate the impact of the focal length tuning, Figure 5 shows two sets of localisation estimates, first with the manually tuned focal length, then with a 50% increase of the focal length. The flight in this example is a stationary flight during which the camera has been moved such that the POI draws a cross and a square in the image frame. The structure present in the estimation of the artificially increased focal length replicates the path drawn by the POI on the image frame. This results from the algorithm being unable to compensate for the displacement of the vPOI in the image frame.

[8]DJI MSDK Documentation KeyCameraZoomRatios

4.6 *Sensitivity to image and telemetry sychronisation*

For a correct localisation, all the data used by the algorithm must be time synchronised. Given that image and telemetry are broadcasted over different channels, they are the most at risk of suffering from synchronisation issues. To verify this, the vertical position in the image frame of the vPOI has been overlayed on the gimbal pitch angle during a stationary flight (Figure 6). It is clear in this example that the gimbal pitch signal is leading the object position in the image signal. Measurement of this delay resulted in approximately 4 samples ($\approx$ 1s) and all the data in this work has been shifted by this amount. The reason for this delay is that images are only timestamped once received by the GCS, but their transfer (including compression and decompression) takes time and this duration can vary depending on the network setup, the traffic on the network and the size or even content of the images. It is unclear whether it is possible to properly timestamp images before sending them, but this would be the best way to measure and compensate for this delay.

4.7 *Gimbal heading measurement*

DJI's MSDK allows the user to access a parameter named *KeyGimbalAttitude*, which according to the SDK's documen-

http://www.imavs.org/

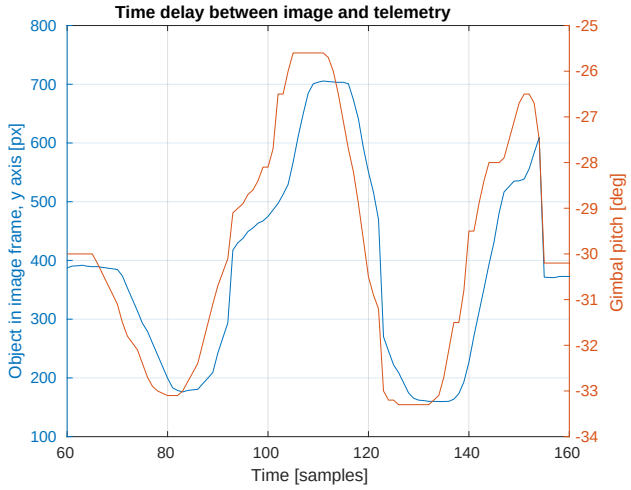


Figure 6: Vertical position in the image frame of the vPOI overlayed on gimbal pitch angle during a stationary flight, before time delay compensation.

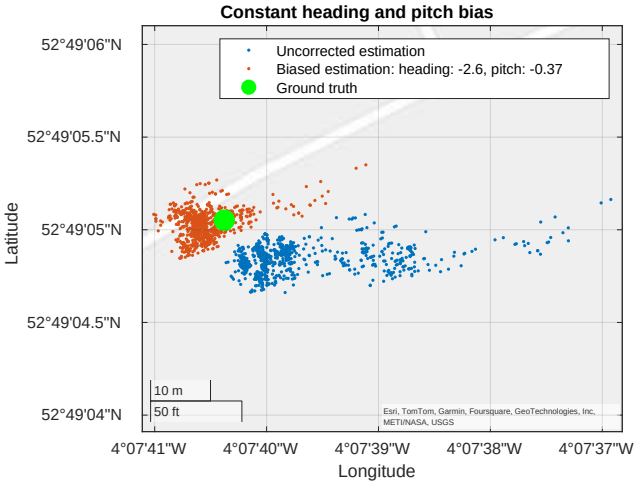


Figure 7: Impact of constant heading and pitch bias on localisation estimate during a 30m altitude flight toward the POI. Biased estimates feature a constant heading and pitch bias and were computed to minimize lateral and longitudinal error respectively.

tation[9] describes the heading of the camera in a NED world frame. However, after inspection of the data, this was not true in certain conditions. Thus, in this work, the drone heading was used (body heading). While this is not the true value, it is a fair approximation since the gimbal heading is controlled to be aligned with the forward direction of the drone and uses this degree of freedom only to smooth out yawing movements. In other words, gimbal heading and drone heading are equal when the drone is not yawing, but their difference may be high during yawing movements.

4.8 Heading and pitch biases

Figure 7 shows the localisation estimate (in blue) for a straight flight towards the POI (in green). Computing the heading bias to minimize the lateral error and the pitch bias to minimize the longitudinal error leads to the biased estimates (in red). The fact that biased estimate collapses to an ellipsoid around the ground truth, suggests that there is indeed a constant error in the heading and pitch measurements. Comparing this bias on different flights unfortunately leads to the conclusion that these biases are not constant across different missions. A possible explanation for this observation is that relative encoders are used in the drone's gimbal, which are well suited for filming since they can be used to stabilise the camera. However, relative encoders are sensitive to their initialization routine and may drift over time if steps are miscounted. This could be mitigated by including a calibration routine in the drone's flight. For example, by localising the pilot holding the remote controller which features a GNSS receiver and thus can be used as a ground truth.

[9]DJI MSDK Documentation KeyGimbalAttitude

4.9 Mission planning recommendation

While the contributions of the error sources listed above could be mitigated by a better understanding of the system or by an improved processing pipeline, they can also be mitigated during mission planning. Hereafter, are three recommendations that can improve the estimation accuracy.

- Ensure that the gimbal pitch is as close to vertical as possible, either by flying higher or closer to the POI (see Section 4.4). This is limited by other requirements, such as image quality, safety distance to the POI or compliance with regulations (altitude ceiling).
- Keep the POI as close to the image centre as possible, thus mitigating the impact of lens distortion and error in the focal length (see Section 4.5). However, this might be hard to achieve during manual flight or even impossible when looking at multiple POIs.
- Make slow movements and maximise stationary observations. This reduces the impact of bad telemetry synchronisation (see Section 4.6).

4.10 Performance evaluation

This section presents two selected flights that can be seen in Figure 8 and 9. Both Figures show the drone's flight path (in red), the estimated position of the POI (in blue) and the ground truth (in green). These flights were selected since they feature two simple movements (moving towards and orbiting around a POI) that are likely to occur during field observations. Moreover, both flights happen at an altitude of 120m above ground, which maximises the performance of the algorithm (see Section 4.4 and 4.9), thus showing how well

http://www.imavs.org/

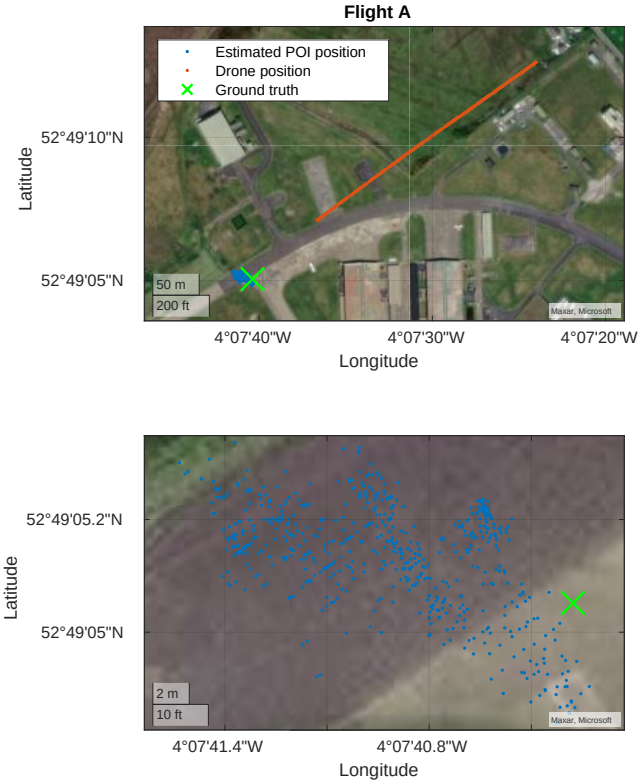


Figure 8: Straight flight towards the POI, altitude: 120m, distance to POI: 385m to 95m, max lateral error: 19.2m, max longitudinal error: 14.4m.

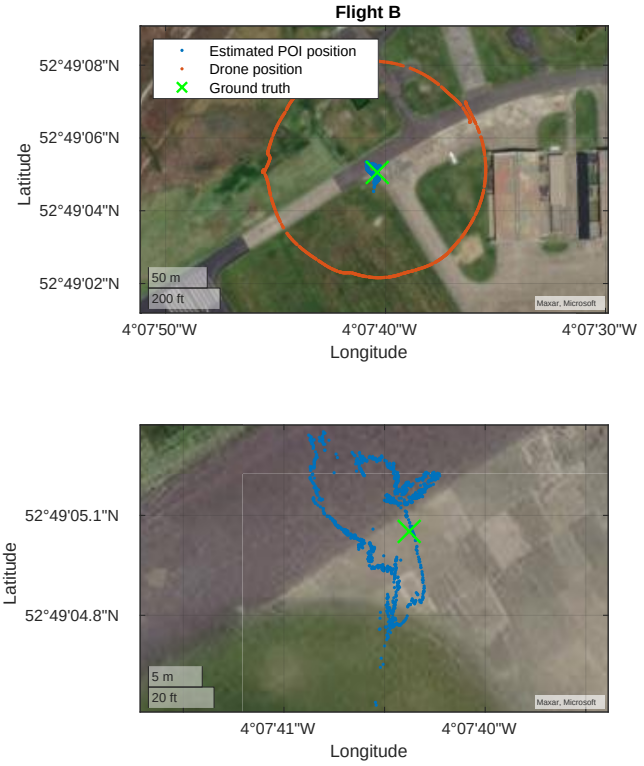


Figure 9: Orbit around the POI, altitude: 120m, distance to POI: 95m, max lateral error: 16.1m, max longitudinal error: 7.4m.

the algorithm can perform in good conditions. Both flights were processed taking into account the tuned focal length discussed in Section 4.5 and the time delay discussed in Section 4.6, but they are not corrected for the constant bias (see Section 7).

For flight A (Figure 8), it is interesting to note that at 385m the error is not larger than 19.2m laterally and 14.4m longitudinally. For flight B (Figure 9), flying at a distance of 95m to the POI, the error is reduced to 16.1m laterally and 7.4m longitudinally. However, there is a lot of structure in the estimated position, suggesting that there are camera movements not properly accounted for. It is likely that this results from the camera heading approximation discussed in Section 4.7.

5 CONCLUSION AND FUTURE WORK

In summary, this work presented a simple localisation algorithm that can be run on a commercial platform: the DJI Mavic 3E. The method has been successfully deployed in the Ol Pejeta Conservancy (Kenya) and at Llanbedr Airfield (Wales). The analysis of the data acquired led to a better understanding of the capabilities and limitations of this algorithm and platform. In particular:

- The estimation error can be reduced by ensuring that the camera is as close to vertical as the mission requirements allow it to be (Section 4.4).
- The estimation is sensitive to the knowledge of the camera focal length (Section 4.5). This can be mitigated by ensuring the POI is in the centre of the image.
- There are synchronisation issues between image and telemetry (Section 4.6).
- There is a constant heading and pitch bias, probably specific to each flight, that should be calibrated out (Section 4.8).

Based on these findings, this work also presented mission planning recommendations to improve the estimation accuracy (Section 4.9).

The resulting localisation error, for a POI 385m away, can be as good as 19.2m laterally and 14.4m longitudinally (Section 4.10).

Further work will address the shortcomings pointed out in the article, such as:

- Develop a more accurate camera model based on a better understanding of the DJI Mavic 3E's camera.
- Address the data synchronisation issue.

http://www.imavs.org/

- Improve the pilot's user interface for better and safer control of the drone.
- Implement a gimbal calibration routine to determine heading and pitch biases.
- Account for terrain elevation during the localisation.
- Filter the estimation over time to improve localization precision.

ACKNOWLEDGEMENTS

This research has been carried out as part of the project WildDrone, funded by the European Union's Horizon Europe Research and Innovation Program under the Marie Skłodowska-Curie Grant Agreement No. 101071224. The WildDrone project has also received funding in part from the EPSRC funded Autonomous Drones for Nature Conservation Missions grant (EP/X029077/1). The authors would like to thank the Kenya Flying Labs, Kenya Civil Aviation Authority, the local Kenya Air Force ATC and the Wildlife Research and Training Institute for their support during the project.

REFERENCES

[1] N. Millner et al. "Exploring the opportunities and risks of aerial monitoring for biodiversity conservation". en. In: *Global Social Challenges Journal* 2.1 (June 2023). Publisher: Bristol University Press Section: Global Social Challenges Journal, pp. 2–23.

[2] M. Hoffmann et al. "The Impact of Conservation on the Status of the World's Vertebrates". In: *Science* 330.6010 (Dec. 2010). Publisher: American Association for the Advancement of Science, pp. 1503–1509.

[3] N. G. Yoccoz, J. D. Nichols, and T. Boulinier. "Monitoring of biological diversity in space and time". English. In: *Trends in Ecology & Evolution* 16.8 (Aug. 2001). Publisher: Elsevier, pp. 446–453. ISSN: 0169-5347.

[4] C. Vermeulen, P. Lejeune, J. Lisein, P. Sawadogo, and P. Bouché. "Unmanned Aerial Survey of Elephants". en. In: *PLOS ONE* 8.2 (Feb. 2013). Publisher: Public Library of Science, e54700. ISSN: 1932-6203.

[5] D. Wang, Q. Shao, and H. Yue. "Surveying Wild Animals from Satellites, Manned Aircraft and Unmanned Aerial Systems (UASs): A Review". en. In: *Remote Sensing* 11.11 (Jan. 2019). Number: 11 Publisher: Multidisciplinary Digital Publishing Institute, p. 1308. ISSN: 2072-4292.

[6] P. Bouché et al. "Game over! Wildlife collapse in northern Central African Republic". en. In: *Environmental Monitoring and Assessment* 184.11 (Nov. 2012), pp. 7001–7011. ISSN: 1573-2959.

[7] Z. Yang et al. "Spotting East African Mammals in Open Savannah from Space". en. In: *PLOS ONE* 9.12 (Dec. 2014). Publisher: Public Library of Science, e115989. ISSN: 1932-6203.

[8] I. Duporge, T. Hodgetts, T. Wang, and D. W. Macdonald. "The spatial distribution of illegal hunting of terrestrial mammals in Sub-Saharan Africa: a systematic map". en. In: *Environmental Evidence* 9.1 (June 2020), p. 15. ISSN: 2047-2382.

[9] R. Lamprey et al. "Comparing an automated high-definition oblique camera system to rear-seat-observers in a wildlife survey in Tsavo, Kenya: Taking multi-species aerial counts to the next level". In: *Biological Conservation* 241 (Jan. 2020), p. 108243. ISSN: 0006-3207.

[10] S. A. Wich and L. Pin Koh. "Mapping". In: *Conservation Drones: Mapping and Monitoring Biodiversity*. Ed. by S. A. Wich and L. P. Koh. Oxford University Press, June 2018, pp. 51–54. ISBN: 978-0-19-878761-7.

[11] L. P. Koh and S. A. Wich. "Dawn of Drone Ecology: Low-Cost Autonomous Aerial Vehicles for Conservation". en. In: *Tropical Conservation Science* 5.2 (June 2012). Publisher: SAGE Publications Inc, pp. 121–132. ISSN: 1940-0829.

[12] Wich. *Drones and Conservation. Drones and Aerial Observation: New Technologies for Property Rights*. en.

[13] E. Bondi et al. "BIRDSAI: A Dataset for Detection and Tracking in Aerial Thermal Infrared Videos". In: 2020, pp. 1747–1756.

[14] D. Chabot and D. M. Bird. "Evaluation of an off-the-shelf Unmanned Aircraft System for Surveying Flocks of Geese". In: *Waterbirds* 35.1 (Mar. 2012). Publisher: The Waterbird Society, pp. 170–174. ISSN: 1524-4695, 1938-5390.

[15] S. Wich, D. Dellatore, M. Houghton, R. Ardi, and L. P. Koh. "A preliminary assessment of using conservation drones for Sumatran orang-utan (*Pongo abelii*) distribution and density". en. In: *Journal of Unmanned Vehicle Systems* 4.1 (Mar. 2016), pp. 45–52. ISSN: 2291-3467.

[16] K. Whitehead et al. "Remote sensing of the environment with small unmanned aircraft systems (UASs), part 2: scientific and commercial applications". en. In: *Journal of Unmanned Vehicle Systems* 02.03 (Sept. 2014), pp. 86–102. ISSN: 2291-3467.

http://www.imavs.org/

[17] J. S. Avila-Sanchez et al. "Evaluating the Use of a Thermal Sensor to Detect Small Ground-Nesting Birds in Semi-Arid Environments during Winter". en. In: *Drones* 8.2 (Feb. 2024). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, p. 64. ISSN: 2504-446X.

[18] M. D. Somers et al. *Airborne point counts: a method for estimating songbird abundance with drones*. en. Pages: 2020.08.15.252429 Section: New Results. Aug. 2020.

[19] W. Andrew et al. "Visual identification of individual Holstein-Friesian cattle via deep metric learning". en. In: *Computers and Electronics in Agriculture* 185 (June 2021), p. 106133. ISSN: 01681699.

[20] J. C. Hodgson et al. "Drones count wildlife more accurately and precisely than humans". en. In: *Methods in Ecology and Evolution* 9.5 (2018). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/2041-210X.12974, pp. 1160–1167. ISSN: 2041-210X.

[21] C. Aubert et al. "Estimating Total Length of Partially Submerged Crocodylians from Drone Imagery". en. In: *Drones* 8.3 (Mar. 2024). Number: 3 Publisher: Multidisciplinary Digital Publishing Institute, p. 115. ISSN: 2504-446X.

[22] R. Viegut et al. "Detection Probability and Bias in Machine-Learning-Based Unoccupied Aerial System Non-Breeding Waterfowl Surveys". en. In: *Drones* 8.2 (Feb. 2024). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, p. 54. ISSN: 2504-446X.

[23] P. Povlsen, D. Bruhn, C. Pertoldi, and S. Pagh. "A Novel Scouring Method to Monitor Nocturnal Mammals Using Uncrewed Aerial Vehicles and Thermal Cameras—A Comparison to Line Transect Spotlight Counts". en. In: *Drones* 7.11 (Nov. 2023). Number: 11 Publisher: Multidisciplinary Digital Publishing Institute, p. 661. ISSN: 2504-446X.

[24] M. Jiménez-Torres, C. P. Silva, C. Riquelme, S. A. Estay, and M. Soto-Gamboa. "Automatic Recognition of Black-Necked Swan (Cygnus melancoryphus) from Drone Imagery". en. In: *Drones* 7.2 (Feb. 2023). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, p. 71. ISSN: 2504-446X.

[25] Q. Ye, J. Luo, and Y. Lin. "A coarse-to-fine visual geo-localization method for GNSS-denied UAV with oblique-view imagery". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 212 (June 2024), pp. 306–322. ISSN: 0924-2716.

[26] Y. Chen and J. Jiang. "An Oblique-Robust Absolute Visual Localization Method for GPS-Denied UAV With Satellite Imagery". In: *IEEE Transactions on Geoscience and Remote Sensing* 62 (2024), pp. 1–13. ISSN: 0196-2892, 1558-0644.

[27] D. Saunders et al. "Radio-tracking wildlife with drones: a viewshed analysis quantifying survey coverage across diverse landscapes". en. In: *Wildlife Research* 49.1 (Feb. 2022). Publisher: CSIRO PUBLISHING, pp. 1–10. ISSN: 1448-5494.

[28] R. Rodriguez, D. M. Jenkins, J. Leary, and R. Perroy. "A direct geolocation method for aerial imaging surveys of invasive plants". en. In: *International Journal of Environmental Science and Technology* (Apr. 2024). ISSN: 1735-2630.

[29] JARUS. *JARUS guidelines on Specific Operations Risk Assessment (SORA)*. May 2024.

http://www.imavs.org/

Bibliography

[1] Ahmad Hammad, Muhammet Demir, and Sophie Armanini. Enhancing perching capabilities of flapping wing robots with silicone-based electromagnetic claws. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 15–22, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-1. 11

[2] Bart Helsen, Thomas Roelandt, Merijn Floren, and Dirk Vandepitte. Flight control of an insect-like tailless flapping wing nano-robot around hover. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 23–32, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-2. 11

[3] Woei-Leong Chan, Hung-Yu Chen, Chia-Le Chin, and Chia-Hsiang Huang. A bio-inspired micro air vehicle and the experimental setup for aerodynamic testing. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 33–40, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-3. 11

[4] Timothy Jakobi, Simon Watkins, Alex Fisher, and Sridhar Ravi. Bumblebees exhibit adaptive flapping responses to air disturbances. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 41–48, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-4. 11

[5] Pablo De Porcellinis Pascau, Jesus Villadangos, Daniel Alaez, and Manuel Prieto. Regulatory compliant uas navigation based on a bitmap grid local path planning algorithm. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 49–55, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-5. 11

[6] Lohithsai Yadala Chanchu, Ruben Zuidgeest, Daphne Stam, and Guido de Croon. Robust heading estimation from polarization images by deep neural networks. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 56–63, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-6. 11

[7] Kevin Guzman Duran, Alejandro Giles, and José Martinez-Carranza. Pose estimation with a lightweight visual-inertial neural network for agile uav flights. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 64–71, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-7. 11

[8] Tim Burgers, Stein Stroobants, and Guido de Croon. Evolving spiking neural networks to mimic pid control for autonomous blimps. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 72–78, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-8. 11

[9] Timothée Gavin, Simon Lacroix, and Murat Bronz. Accelerating swarms : Harnessing hardware acceleration and parallelization in multi-agent reinforcement learning. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 79–86, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-9. 11

[10] Mauro Villanueva Aguado, Christophe De Wagter, and Guido de Croon. Adaptive neural network quadrotor trajectory tracking controller tolerant to propeller damage. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 87–99, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-10. 11

[11] Aldrich Alfredo Cabrera Ponce, Leticia Oyuki Rojas Pérez, Manuel Isidro Martin Ortiz, and José Martinez-Carranza. Binary networks and continual learning for pose estimation from a single aerial image. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 100–108, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-11. 11

[12] Burhan Saeed and Peter Thomas. Feasibility study of a solar powered hybrid micro-aerial vehicle. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 109–115, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-12. 11

http://www.imavs.org/

[13] Diego Alexis Martínez-Velasco, Marco Antonio Martínez-Ramírez, and Hugo Rodríguez-Cortés. A nonlinear autopilot for a piloted quad-plane. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 116–123, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-13. 11

[14] Daniel Olivares-Figueroa, Israel Cruz-Vega, Alejandro Gutierrez-Giles, and José Martinez-Carranza. Spherical aerial manipulator robot for exploration in complex forest environments. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 124–130, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-14. 11

[15] Chantelle Singh, Shahab Kazemi, Joao Buzzatto, and Karl Stol. Dexterity assessment of fully-actuated uavs. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 131–138, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-15. 11

[16] Edgar Ulises Rojo Rodríguez, Erik Gilberto Rojo Rodríguez, Sergio Araujo Estrada, and Octavio García Salazar. Design of a tapered-wing tiltrotor uav for multi-mission applications. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 139–147, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-16. 11

[17] Nicholas Kay, Peter Richards, Karl Stol, Rory Buchanan, and Jérémie Bannwarth. Effect of aerodynamic summation on multirotor uav hover performance. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 148–155, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-17. 11

[18] Sebastian Kean, Matthew Marino, Simon Watkins, and Abdulghani Mohamed. Swarm based airborne wind measurement. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 156–162, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-18. 11

[19] Victoria Vazquez and José Martinez-Carranza. Landing zone detection for mavs using depth images and vision transformers. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 163–170, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-19. 11

[20] Caleb Rascon, L. Oyuki Rojas-Perez, Victoria Eugenia Vazquez-Meza, Aldrich A. Cabrera-Ponce, and José Martinez-Carranza. Crossing gate detection using audio frequency pitch tracking. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 171–176, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-20. 11

[21] Leandro Goncalves dos Santos, João Vitor Nascimento Souza, Durval Ferreira Sobrinho Junio, Rafael Gomes de Oliveira, Savio Pessôa Afonso, Jeovana Miranda Souza, Fábio Santos Lima, and Reinaldo Monteiro Cotrim. Comparative analysis of cameras for aruco marker recognition in unmanned aerial vehicles. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 177–184, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-21. 11

[22] Javier Melero-Deza, Rafael Perez-Segui, Pedro Arias-Perez, Miguel Fernandez-Cortizas, David Perez-Saura, Guillermo Gp-Lenza, Miguel Tradacete-Agreda, Carlos Santos Pérez, Francisco Javier Rodríguez Sánchez, and Pascual Campoy. Photovoltaic plant monitoring and inspection through synergic integration of uavs and iot. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 185–192, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-22. 11

[23] Till M. Blaha, Ewoud J.J. Smeur, Bart D.W. Remes, and Coen C. de Visser. Flying a quadrotor with unknown actuators and sensor configuration. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 193–199, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-23. 11

[24] Jiaxun Leng, Steven Ng, and Shun Tang. Novel flapping wing micro aerial vehicle design featuring a bio-inspired abdomen mechanism. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 200–208, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-24. 11

[25] Ivan Panov, Jes Jepsen, Mirko Presser, and Kjeld Jensen. A method for determining the safe separation to an unknown-model uncooperative uav in u-space. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 209–218, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-25. 11

[26] Kjeld Jensen, Klavs Andersen, and Jes Jepsen. A detect and avoid concept for uas operations near hospital emergency medical services helipads in uncontrolled urban airspace. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 219–227, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-26. 11

http://www.imavs.org/

[27] Mads Fogh Høffer, Victor Dahl Herlev, Guy Maalouf, Jes Jepsen, Erik Winkler, Ivan Panov, and Kjeld Jensen. Extended reception coverage and spoofing localization of direct remote id utilizing ad-hoc mesh networking. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 228–235, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-27. 11

[28] Nicholas Carlier, Trista Yuting Ma, Nicholas Kay, Amir Pirooz, and Cesar Azorin-Molina. Effects of simultaneous turbulence and ice accretion on small fixed-wing uav performance. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 236–242, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-28. 11

[29] Ivan Panov, Mouad Boumediene, Henrik Skov Midtiby, and Kjeld Jensen. Gen4jectory algorithm – 4-d trajectory planning with minimised flight time for multiple rotary-wing uavs. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 243–253, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-29. 11

[30] Armand-Ioan Curpanaru, Fabrice Demourant, and Florian Sansou. Control of the darko tail-sitter drone through an lmi-based static output feedback design. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 254–261, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-30. 11

[31] Chantelle Singh, Shahab Kazemi, Jonty Kirk, Katrina Chan, and Karl Stol. A study of gain tuning for prolonged physical contact of fully-actuated uavs. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 262–269, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-31. 11

[32] Amelie Carrierou, Cedric Join, and Jean-Philippe Condomines. Restricted model control applied to the quanser aero with uncertain parameters. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 270–276, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-32. 11

[33] Pieter Van Holm, Apoorva Busunur Mallikarjuna, Carmelo Simon Soria, and Leandro Ribeiro Lustosa. On the lateral non-stabilizability of pid controllers for vision-based line-following multicopters through roll angle setpoints. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 277–284, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-33. 11

[34] João Aires Marsicano and Agnes Almeida. Swarm-in-blocks: Simplifying drone swarm programming with block-based language. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 285–292, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-34. 11

[35] Vitor Garcia Ribeiro, André Carmona Hernandes, Guilherme Barros Villela, and Marcelo Becker. Comparative analysis of esc and flight controller communication protocols and their impact on brushless motor response and drone control. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 293–300, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-35. 11

[36] Vandita Shukla, Luca Morelli, Pawel Trybala, Fabio Remondino, Wentian Gan, Yifei Yu, and Xin Wang. Exploring the potential of collaborative uav 3d mapping in kenyan savanna for wildlife research. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 301–308, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-36. 11

[37] Rhiannon Elliott-Roe, Kieran Wood, and Özgün Ozer. Effects of wind speed on quadcopter ceiling effect. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 309–315, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-37. 11

[38] Edouard Rolland, Kasper Grøntved, Lucie Laporte-Devylder, Jenna Kline, Ulrik Lundquist, and Anders Christensen. Drone swarms for animal monitoring: A method for collecting high-quality multi-perspective data. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 316–323, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-38. 11

[39] Kilian Meier, Arthur Richards, Matthew Watson, Caspian Johnson, Duncan Hine, Tom Richardson, and Guy Maalouf. Wildbridge: Conservation software for animal localisation using commercial drones. In T. Richardson, editor, *15th annual International Micro Air Vehicle Conference and Competition*, pages 324–333, Bristol, United Kingdom, Sep 2024. Paper no. IMAV2024-39. 11

http://www.imavs.org/