

http://www.imavs.org/

IMAV 2022

13th International Micro Air Vehicle Conference

September 12-16, 2022

Delft, the Netherlands

www.imavs.org

Dronisos bitcraze Robo Crops U.S.ARMY DEVCOM

TU Delft | AgTech Institute TU Delft | Robotics Institute

UNMANNED VALLEY

GEMEENTE WESTLAND

Office of Naval Research GLOBAL Science & Technology

FLAPPER DRONES

Contents

IMAV2022 **1**
Preface . . . 3
International IMAV Committee . . . 4
Program Committee . . . 4
Organizing Committee . . . 4
Sponsors . . . 5
Call for Papers . . . 6
Competitions . . . 7
Sessions . . . 8
Keynote Speakers . . . 9
List Of Papers . . . 13
Authors . . . 14

Papers **16**
Lateral guidance and control for a fixed-wing aircraft . . . 17
Panel Method Based Guidance for Fixed Wing Micro Aerial Vehicles . . . 25
Reconnaissance Target Grouping and Assignment Considering Fault Severity for UAV Swarm . . . 32
Evaluation of drag coefficient for a quadrotor model . . . 39
Improving the computational efficiency of ROVIO . . . 47
Fast NIR-Single-Pixel-Imaging enhancement under scattering environment . . . 53
Handling Pitch Variations for Visual Perception in MAVs: Synthetic Augmentation and State Fusion . . . 59
Using trajectory oscillation timing improves in-flight odometry based solely on optic flows . . . 66
Seeing with sound – surface detection and avoidance by sensing self-generated noise . . . 75
Deep Learning-based Flight Speed Estimation using Thermal Anemometers . . . 84
Semantic Segmentation of RGB Aerial Images Using CNN and Superregions . . . 92
Multi-Model Continual Learning for Camera Localisation from Aerial Images . . . 103
Leveraging a Neural Pilot via Automatic Gain Tuning using Gate Detection for Autonomous Drone Racing . . . 110
INDI Control for the oblique wing-quad plane drone . . . 119
Miniaturization and Control of an Unmanned Tiltwing Aircraft . . . 127
Attitude control for a tiltwing aircraft under tail actuator failures . . . 136
Robust flight control using Hybrid Incremental Nonlinear Dynamic Inversion Control for Tiltrotor UAV . . . 144
Recyclable bio-based composite Flax/Elium for UAV applications . . . 152
Development of a Mission-Tailored Tail-Sitter MAV . . . 159
C wing geometric redesign based on aircraft flying qualities . . . 169
Micro-drone autopilot architecture for efficient static scheduling . . . 175
Hover Controller Design and Implementation for a Dragonfly-like Flapping Wing . . . 183
Design and Joint Control of a Conjoined Biplane and Quadrotor . . . 193
Total Energy Control System Design for Longitudinal Dynamics of Complex Articulated Ornithopter . . . 201
Frequency based strategy for hybrid-powered unmanned aerial vehicle . . . 207

http://www.imavs.org/

Preface

On behalf of the Local Organizing Committee, it is our pleasure to present the proceedings of the 13^{th} International Micro Air Vehicle Conference, which was held in Delft, the Netherlands from September 12-16, 2022. For the fourth time, the IMAV was organized by the Micro Air Vehicle Lab of the Delft University of Technology in the Netherlands.

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. This year of 2022, was very special as it forms the first real-life event since the COVID19 pandemic. While, very unfortunately, several restrictions still prohibited several teams to participate, we are delighted to see over 140 full participants and over 100 visitors from industry. In particular, the greenhouse challenge attracted a lot of visitors from industry and students in the agricultural domain, which is a measure for the expected value of drones' added value to this field.

These proceedings contain twenty-five peer-reviewed scientific papers by sixty-seven authors organized in six sessions presented at the IMAV in 2022. 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. Five articles were nominated for the best paper award, out of which one was granted the "Best Paper Award" at the award ceremony. Based on the quality of the scientific and technical contribution, ten 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 thirteen members of the International Committee from nine institutes who guide IMAV overall, and the seventeen reviewers from nine institutes, who played an important role in assuring the quality of all the papers. In addition to the presentation of the scientific papers, four 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-2022. Last but not least, we are particularly grateful for the sponsorship provided by our wide range of sponsors: Gemeente Westland, RoboCrops, Bitcraze AB, Unmanned Valley, Office of Naval Research, US Army, Dronisos - Drone Light Shows, Flapper Drones, TU Delft Robotics Institute, and TU Delft AgTech Institute.

Delft, the Netherlands. September 2022

G. de Croon and C. De Wagter
MAVLab, TUDelft

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
Mohamed Abdulghani . . . . . RMIT, Melbourne, Australia
Murat Bronz . . . . . ENAC, Toulouse, France
Pascual Campoy . . . . . UPM, Madrid, Spain
Simon Watkins . . . . . RMIT, Melbourne, Australia

Program Committee

Bart Remes . . . . . TUDelft, Delft, the Netherlands
Bas Van Oudheusden . . . . . TUDelft, Delft, the Netherlands
Ben M. Chen . . . . . CUHK, Hong-Kong, China
Christophe De Wagter . . . . TUDelft, Delft, the Netherlands
Dieter Moormann . . . . . RWTH, Aachen, Germany
Ewoud Smeur . . . . . TUDelft, Delft, the Netherlands
Gautier Hattenberger . . . . . ENAC, Toulouse, France
Guido de Croon . . . . . TUDelft, Delft, the Netherlands
Hans Goosen . . . . . 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
Mohamed Abdulghani . . . . . RMIT, Melbourne, Australia
Murat Bronz . . . . . ENAC, Toulouse, France
Pascual Campoy . . . . . UPM, Madrid, Spain
Salua Hamaza . . . . . TUDelft, Delft, the Netherlands
Simon Watkins . . . . . RMIT, Melbourne, Australia

Organizing Committee

Guido de Croon, ... Organizer
Bart Remes . . . . . Organizer
Christophe De Wagter . . . . . Organizer
Ewoud Smeur . . . . . Organizer
Salua Hamaza . . . . . Organizer
Julien Dupeyroux . . . . . Organizer
Alessandro Mancinelli . . . . . Staff
Anique Altena . . . . . Staff
Anton Bredenbeck . . . . . Staff
David Wielemaker . . . . . Staff
Federico Paredes-Valles . . . . . Staff
Freek van Tienen . . . . . Staff
Hang Yu . . . . . Staff
Jack Hayton . . . . . Staff
Jesse Hagenaars . . . . . Staff
Kitso Epema . . . . . Staff
Liming Zheng . . . . . Staff
Matthew Yedutenko . . . . . Staff
Nicholas Dvorsky . . . . . Staff
Salil Luesutthiviboon . . . . . Staff
Stavrow Bahnam . . . . . Staff
Stein Stroobants . . . . . Staff
Sunyi Wang . . . . . Staff
Sunyou Hwang . . . . . Staff
Sven Pfeiffer . . . . . Staff
Tom van Dijk . . . . . Staff
Yilun Wu . . . . . Staff
Ziqing Ma . . . . . Staff

http://www.imavs.org/

Sponsors













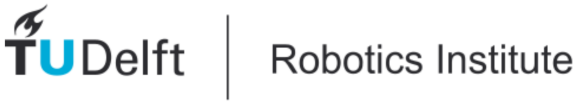














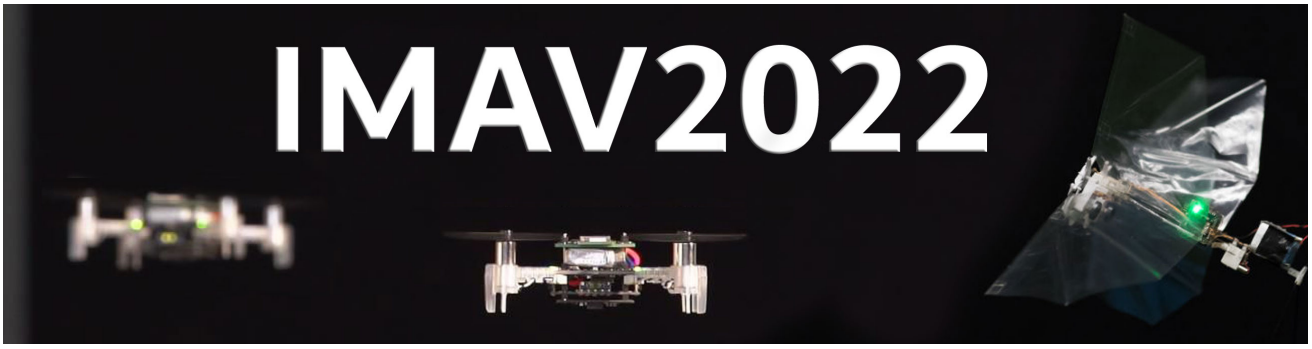


Websites

- https://www.dronisos.com/
- https://www.bitcraze.io/
- https://unmannedvalley.nl/
- https://www.gemeentewestland.nl/
- https://www.tudelft.nl/en/agtech-institute
- https://tudelftroboticsinstitute.nl/
- https://www.army.mil/devcom
- https://www.nre.navy.mil/
- https://robocrops.tech/
- https://flapper-drones.com/

http://www.imavs.org/

Call for Papers



The IMAV conference focuses on all scientific studies related to Micro Air Vehicles (MAVs), ranging from aerodynamics and design of small drones to their control and artificial intelligence. The conference is always combined with real-world competitions that aim to bring theory into practice and advance the state of the art. IMAV 2022 will be organized by TU Delft.

Submission Guidelines

Articles will be fully peer-reviewed and published open access. A selection of best-reviewed articles will be invited to special issues in the International Journal of Micro Air Vehicles and in the Unmanned Systems.

All papers must be original and not simultaneously submitted to another journal or conference. They must follow the template and be about 6 pages in length. File sizes should be minimized (max 5MB) while graphical quality should be maximized by using vector graphics whenever possible.

List of Topics

We solicit scientific articles in all areas related to Micro Air Vehicles, for example on the following (non-exhaustive) list of topics:

- (Unsteady) aerodynamics at low Reynolds numbers
- Agile/fast autonomous flight (autonomous drone racing
- Hybrid drones (combining fixed wings and rotors)
- Design of novel drone types (flapping-wing drones / silent drones)
- Biomimetic/bio-inspired drones for outdoors and indoors
- AI-capable drones - running (spiking) deep nets
- Drones that interact robustly with their environment (aerial manipulation)
- New control methods for drones
- Swarming (e.g., collaborative carrying, exploration
- Autonomous navigation in GPS-denied environments
- Integration of drones in airspace (sense-and-avoid, UTM, ...)
- Drone applications (search-and-rescue, greenhouse, package delivery)

Venue

- Location: Delft, the Netherlands (+ online)
- Event dates: September 12 -16, 2022. Web site: http://www.imavs.org/2022
- Article submission deadline: May 15, 2022

Contact

All questions about submissions should be emailed to imavsdotorg@gmail.com

http://www.imavs.org/

Competitions



Nanocopter Challenge



Greenhouse Challenge



Package Delivery Challenge

http://www.imavs.org/

Conference Oral Presentations

Guidance and Control

Lateral guidance and control for a fixed-wing aircraft . . . p.17
Panel Method Based Guidance for Fixed Wing Micro Aerial Vehicles . . . p.25
Reconnaissance Target Grouping and Assignment Considering Fault Severity for UAV Swarm . . . p.32
Evaluation of drag coefficient for a quadrotor model . . . p.39

Perception

Improving the computational efficiency of ROVIO . . . p.47
Fast NIR-Single-Pixel-Imaging enhancement under scattering environment . . . p.53
Handling Pitch Variations for Visual Perception in MAVs: Synthetic Augmentation and State Fusion . . . p.59
Using trajectory oscillation timing improves in-flight odometry based solely on optic flows . . . p.66
Seeing with sound – surface detection and avoidance by sensing self-generated noise . . . p.75

Deep Learning

Deep Learning-based Flight Speed Estimation using Thermal Anemometers . . . p.84
Semantic Segmentation of RGB Aerial Images Using CNN and Superregions . . . p.92
Multi-Model Continual Learning for Camera Localisation from Aerial Images . . . p.103
Leveraging a Neural Pilot via Automatic Gain Tuning using Gate Detection for Autonomous Drone Racing . . . p.110

Control of Transitioning MAV

INDI Control for the oblique wing-quad plane drone . . . p.119
Miniaturization and Control of an Unmanned Tiltwing Aircraft . . . p.127
Attitude control for a tiltwing aircraft under tail actuator failures . . . p.136
Robust flight control using Hybrid Incremental Nonlinear Dynamic Inversion Control for Tiltrotor UAV . . . p.144

MAV Design

Recyclable bio-based composite Flax/Elium for UAV applications . . . p.152
Development of a Mission-Tailored Tail-Sitter MAV . . . p.159
C wing geometric redesign based on aircraft flying qualities . . . p.169
Micro-drone autopilot architecture for efficient static scheduling . . . p.175

Design and Control

Hover Controller Design and Implementation for a Dragonfly-like Flapping Wing . . . p.183
Design and Joint Control of a Conjoined Biplane and Quadrotor . . . p.193
Total Energy Control System Design for Longitudinal Dynamics of Complex Articulated Ornithopter . . . p.201
Frequency based strategy for hybrid-powered unmanned aerial vehicle . . . p.207

http://www.imavs.org/

Keynote Speakers

Hyper-dimensional Active Perception: a journey into the mind of a robotic bee

Yiannis Aloimonos, Computer Vision Laboratory – University of Maryland, College Park

Abstract



Action and perception are often kept in separated spaces, which is a consequence of traditional vision being frame-based and only existing in the moment and motion being a continuous entity. This bridge is crossed by the dynamic vision sensor (DVS), a neuromorphic camera that can see the motion. We propose a method of encoding actions and perceptions together into a single space that is meaningful, semantically informed, and consistent by using hyper-dimensional binary vectors (HBVs). We show that the visual component can be bound with the system velocity to enable dynamic world perception, which creates an opportunity for real-time navigation and obstacle avoidance with active perception. Actions performed by an agent are directly bound to the perceptions experienced to form its own "memory." Furthermore, because HBVs can encode entire histories of actions and perceptions—from atomic to arbitrary sequences—as constant-sized vectors, auto-associative memory was combined with deep learning paradigms for controls. Using this methodology we can implement, for manoquadrotors with all processing on board, a hierarchy of sensorimotor loops providing a set of competences (egomotion, moving object detection, obstacle avoidance, homing and landing) which can be interfaced with episodic, procedural and semantic memory, giving rising to a minimal cognitive system. The talk will conclude with a journey into the mind of a robotic bee using a variety of experimental results.

Biography

Yiannis Aloimonos is a Professor of Computational Vision and Intelligence at the Department of Computer Science, University of Maryland, College Park, and the Director of the Computer Vision Laboratory at the Institute for Advanced Computer Studies (UMIACS). He is also affiliated with the Institute for Systems Research, the Neural and Cognitive Science Program and the Maryland Robotics Center. He was born in Sparta, Greece and studied Mathematics in Athens and Computer Science at the University of Rochester, NY (PhD 1990). He is interested in Active Perception and the modeling of vision as an active, dynamic process for real-time robotic systems. For the past five years he has been working on bridging signals and symbols, specifically on the relationship of vision to control, and the relationship of action and language using Hyper-dimensional Computing.

http://www.imavs.org/

Bio-inspired aerodynamics, actuation, sensing and control

Richard Bomphrey – Professor of Comparative Biomechanics – Royal Veterinary College

Abstract

How can the observation of animal aerodynamics inform bio-inspired aircraft? I will discuss how fundamental bioscience research can reach outside animal ecology and physiology to offer solutions to engineering challenges.



Flying animals must perceive and avoid obstacles, often in environments deprived of visual sensory cues. In my first example, I will show how collision-avoidance in nocturnal mosquitoes can be mediated by mechanosensory feedback, based on modulations of their own induced aerodynamic and acoustic fields as they enter ground- or wall-effect. Our computational fluid dynamics and aeroacoustic simulations are derived from detailed wing kinematics extracted from high-speed recordings of freely flying Culex quinquefasciatus mosquitoes. Based on Culex data, we have built palm-sized prototypes carrying a suite of bio-inspired sensor packages. I will discuss how aerodynamic and aeroacoustic cues that are associated with close proximity to the ground and wall planes could provide useful information to the flight controller (a mechanism we term 'aerodynamic imaging') and how mosquitoes perceive and interact with others using acoustic interactions.

In my second example, I will present our work based on measuring the changing shape of birds in flight. I will show how they minimise drag in a different way from conventional aircraft design on account of their smaller size, and how they remain unperturbed by strong gusts. Our detailed three-dimensional reconstructions of surface geometries show how wing elevation around the shoulder joint acts as an initially passive suspension system that rejects gusts, steadying the payload. The mechanism works most effectively when the aerodynamic centre of pressure is aligned with the mechanical centre of percussion, and therefore can be tuned either by changing wing shape or by the distribution of mass within the wing. Together, these topics give a small glimpse into the potential that continues to exist within the natural world for the improvement of small aircraft.

Biography

Richard's research blends biology and engineering. He uses biomechanics as a tool to investigate evolutionary biology and how the physical environment determines the morphology and control systems of flying animals. He has worked on the sensory mechanisms of insects and birds, including flow-sensing, load-sensing, and optic flow. Richard's work uses advanced equipment to investigate animal flight and understand their aerodynamic footprints by observing the motion of smoke or bubbles floating in the air. He has applied insights from biology to aerial robots inspired by birds and insects. Richard joined the Structure and Motion Laboratory at the Royal Veterinary College, University of London, in 2013 after reading Biological Sciences at Exeter, undertaking a DPhil (PhD) in Oxford, postdoctoral positions in Oxford and Bath, and an EPSRC Fellowship. He is currently Professor of Comparative Biomechanics at the Royal Veterinary College and Interim Vice Principal for Research.

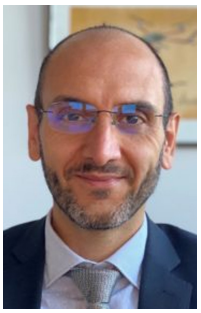
http://www.imavs.org/

Is control a solved problem in robotics MAV research?

Antonio Franchi – Professor in Aerial Robotics Control – Twente University

Abstract

In these days MAV and robotics research in general are dominated by data-driven approaches that are mainly focused on scientific problems related to perception and other AI-related problems. The design, modeling, and control problems, which have been the backbone of MAV research in the past decades seem to suffer a period of decline in terms of popularity. There are many possible explanations for this phenomenon. Perhaps there are no more relevant open control problems for UAVs; or maybe still unsolved problems are now perceived as solved because of some sort of 'mass hallucination'; or perhaps the attention is now distracted by other problems which have been neglected in the past because too hard to solve and now seem/are finally within reach thanks to the new AI wave; or another explanation could be that the problems still open in control are so difficult that everybody pretend to ignore them, like the 'Fox and the Grapes'. Or the motivation could be a different one of course. Or perhaps there is no decline in popularity at all and this research field is actually in very good shape but just not advertised enough to emerge.



In this keynote I will give my historical and technical point of view on the question whether there are still significant open scientific problems in UAV research on the design and control side and, in case the answer is yes, I will try to critically rank such open problems in terms of difficulty and provide possible technical directions to explore the paths toward their solutions.

Biography

Antonio Franchi (https://homepages.laas.fr/afranchi/robotics/) is a Full Professor in Aerial Robotics Control at the University of Twente, The Netherlands, in the Robotics and Mechatronics department, and an associated researcher at LAAS-CNRS (RIS team), Toulouse, France. He is a IEEE Senior Member.

From 2014 to 2019 he was a Permanent Researcher at CNRS and the leader of the aerial robotics activities at LAAS-CNRS. From 2010 to 2013 he was a Research Scientist and then a Senior Research Scientist at the Max Planck Institute for Biological Cybernetics in Germany, and the scientific leader of the group "Autonomous Robotics and Human Machine Systems". He received the Laurea (M.Sc.) degree (summa cum laude) in Electronic Engineering and the Ph.D. degree in System Engineering (Feb. 2010) from Sapienza University of Rome, Italy. In 2009 he was a visiting student at the University of California at Santa Barbara.

His main research interests lie in the robotics area, with a special regard to control and estimation problems and applications ranging across motion and physical interaction control, decentralized control/estimation/coordination, haptics, and hardware/software architectures. His main areas of expertise are aerial robotics and multiple-robot systems.

He published more than 150 papers in international journals, books, and conferences and gave more than 90 invited talks in international venues since 2010. In 2010 he was awarded with the "IEEE RAS ICYA Best Paper Award" for one of his works on Multi-robot Exploration. In 2018 he was a recipient of the 2018 IEEE RAS Most Active Technical Committee Award.

He was Associate Editor of the IEEE Transactions on Robotics from 2016 until 2021 and he is Senior Editor for IEEE ICRA since 2020. He has been associate editor of the IEEE Robotics & Autom. Mag. (2013 to 2016), IEEE ICRA (2014 to 2019), IEEE/RSJ IROS (2014 to 2017) and the IEEE Aerospace and Electric Systems Magazine (2015).

He is the project coordinator of JCJC ANR MuRoPhen, co-coordinator of the FlyCrane project, and a was a contributor to PRO-ACT H2020, all focused on multi-robot coordination/manipulation. He is the local coordinator of the EU H2020 Aerial-CORE project, he has been local coordinator in the EU H2020 AEROARMS project, the creator of the ANR PRC 'The Flying Co-worker' project, and he contributed to the EU FP7 ARCAS project, all focused on aerial robotic manipulation.

He is the co-founder and co-chair of the IEEE RAS Technical Committee on Multiple Robot Systems (>450 members).

He co-funded and was the program co-chair of the IEEE-sponsored International Symposium on Multi-robot and Multi-agent Systems (MRS 2017, 2019, and 2021). He co-organized the IEEE-RAS-sponsored 2019 and 2016 Summer Schools on Multiple Robot Systems at NUS, Singapore and CTU, Prague, and more than 15 workshops on Multi-robot Systems, Aerial Robots and Teleoperation at IEEE ICRA, IEEE/RSJ IROS, and RSS, among others.

Since 2010, he has been the mentor of 11 graduated PhD student and he is currently mentoring 8 PhD candidates. One the PhD theses he mentored won the Best Robotics Thesis French National Award in 2019.

http://www.imavs.org/

Storytelling with drone swarms

Laurent PERCHAIS, CEO – Rémi POTET & Corentin GIRAUD, Drone Engineers at DRONISOS

Abstract



Dronisos, with its unrivalled technology developed completely in-house, has turned the sky into the world's biggest 3-dimensional screen using thousands of drones as animated pixels.

Major Events, Global Brands and world-renowned Theme Parks have all adopted this new medium of communication and entertainment, making Dronisos a worldwide leader in drone entertainment with over 15,000 performances to date.

Dronisos has invented a new medium for storytellers to bring their stories to life on the world's biggest stage – the sky.

However, for this keynote, we want to tell you OUR story. The story of a 5-year journey a team of engineers passionate about drones took on, inventing a new medium of expression while tackling technical and scientific challenges of autonomous drone swarms along the way.

Biography

Laurent PERCHAIS is the CEO of Dronisos. He is a graduate of the Ecole Polytechnique (Paris, France) and HEC Business School (Paris, France). Following his first role at Canal+ (French TV player), he held various positions at Orange (French Telco) both as an Engineer and as a Business manager, where he ended as Head of Strategy for Orange Content (TV, Music, Gaming. . . .) before becoming Dronisos CEO in 2017.

Remi POTET is a former IMAV participant and now a Dronisos expert in drone engineering. He holds an MSc from ISAE SUPAERO(Toulouse, France) and has been part of Dronisos' journey from almost the beginning (2018).

Corentin GIRAUD is a drone industry veteran. After a successful experience at Azur Drones, he joined Dronisos in 2020 to lead the development of Dronisos in the Defense & Safety sector. He graduated from ISAE-SUPAERO (Toulouse, France).

http://www.imavs.org/

Papers

[1] Lateral guidance and control for a fixed-wing aircraft . . . p.17
[2] Panel Method Based Guidance for Fixed Wing Micro Aerial Vehicles . . . p.25
[3] Reconnaissance Target Grouping and Assignment Considering Fault Severity for UAV Swarm . . . p.32
[4] Evaluation of drag coefficient for a quadrotor model . . . p.39
[5] Improving the computational efficiency of ROVIO . . . p.47
[6] Fast NIR-Single-Pixel-Imaging enhancement under scattering environment . . . p.53
[7] Handling Pitch Variations for Visual Perception in MAVs: Synthetic Augmentation and State Fusion . . . p.59
[8] Using trajectory oscillation timing improves in-flight odometry based solely on optic flows . . . p.66
[9] Seeing with sound – surface detection and avoidance by sensing self-generated noise . . . p.75
[10] Deep Learning-based Flight Speed Estimation using Thermal Anemometers . . . p.84
[11] Semantic Segmentation of RGB Aerial Images Using CNN and Superregions . . . p.92
[12] Multi-Model Continual Learning for Camera Localisation from Aerial Images . . . p.103
[13] Leveraging a Neural Pilot via Automatic Gain Tuning using Gate Detection for Autonomous Drone Racing . . . p.110
[14] INDI Control for the oblique wing-quad plane drone . . . p.119
[15] Miniaturization and Control of an Unmanned Tiltwing Aircraft . . . p.127
[16] Attitude control for a tiltwing aircraft under tail actuator failures . . . p.136
[17] Robust flight control using Hybrid Incremental Nonlinear Dynamic Inversion Control for Tiltrotor UAV . . . p.144
[18] Recyclable bio-based composite Flax/Elium for UAV applications . . . p.152
[19] Development of a Mission-Tailored Tail-Sitter MAV . . . p.159
[20] C wing geometric redesign based on aircraft flying qualities . . . p.169
[21] Micro-drone autopilot architecture for efficient static scheduling . . . p.175
[22] Hover Controller Design and Implementation for a Dragonfly-like Flapping Wing . . . p.183
[23] Design and Joint Control of a Conjoined Biplane and Quadrotor . . . p.193
[24] Total Energy Control System Design for Longitudinal Dynamics of Complex Articulated Ornithopter . . . p.201
[25] Frequency based strategy for hybrid-powered unmanned aerial vehicle . . . p.207

http://www.imavs.org/

Authors

A

Amos, Stephen p.75

B

Bahlouli, Nadia p.152
Bahnam, Stavrow p.47
Bartoli, Nathalie p.159
Bergantin, Lucia p.66
Bernabe-Lorranca, Beatriz p.169
Bilgin, Zeynep p.25
Bomphrey, Richard J p.75
Bonneval, Fabien p.175
Bronz, Murat p.25, p.39, p.159

C

Cabrera-Ponce, Aldrich Alfredo p.103
Cereda, Elia p.59
Condomines, Jean-Philippe p.39
Coquet, Charles p.66
Croon, Guido de p.47

D

Durini, Daniel p.53

F

Fernandez, Luiz Fernando Tiberio p.159

G

Giusti, Alessandro p.59
Grolleau, Emmanuel p.175

H

Hattenberger, Gautier p.39, p.175

J

Jeaong, Hoijo p.144

K

Kiefer, Renaud p.152, p.207
Kim, Inrae p.201
Kim, Jingu p.32
Kim, Seungkeun p.32, p.144, p.201
Kim, Taegyun p.32

L

Ladeira, Matheus p.175
Laroche, Edouard p.207
Lashgari, Mohammad p.183
Lefebvre, Martin p.152, p.207
Lefebvre, Thierry p.159

M

Magdaleno, José de Jesús p.53
Marchand, Nicolas p.66
Mariana-Edith, Miranda-Varela p.92
Martin-Ortiz, Manuel Isidro p.103
Martinez-Carranza, Jose p.53, p.92, p.103, p.110
Martinez-Ramirez, Marco A. p.17
Mesbahi, Tedjanie p.207
Moormann, Dieter p.127
Morin, Pascal p.84
Müller, Julian p.127

N

Naghash, Abolghasem p.183
Negre, Amaury p.66

O

Oh, Woongtaek p.201
Ouhammou, Yassine p.175

P

Palossi, Daniele p.59
Pavot, Thomas p.152, p.207

Q

Qu, Jingang p.84
Quero, Carlos Alexander p.53

http://www.imavs.org/

R

Raharijaona, Thibaut p.66
Ramos-Garcia, Ruben p.53
Remes, Bart D.W. p.119, p.193
Rodríguez-Cortés, Hugo p.17, p.169
Rojas-Perez, L. Oyuki p.110
Rojas-Perez, Leticia Oyuki p.92
Ruffier, Franck p.66

S

Schatten, Daniel p.136
Schröter, Shawn p.193
Seo, Ilwon p.32
Smeur, Ewoud J.J. p.193
Suk, Jinyoung p.144, p.201

V

Vedrines, Marc p.152

W

Wagter, Christophe de p.47
Wang, Ze p.84
Wijngaarden, Dennis van p.119
Wilshin, Simon p.75

Y

Yavrucuk, Ilkay p.25

http://www.imavs.org/

http://www.imavs.org/

Release

Released on September 12^{th} 2022.

V1.1, Rev.16827, 2026-10-18 17:49
Created by: C. De Wagter

Lateral guidance and control for a fixed-wing aircraft.

Martínez-Ramírez Marco A.*, Rodríguez-Cortés H.*, and Corona-Sánchez José J.†
*Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional,
Sección de Mecatrónica, Departamento de Ingeniería Eléctrica,
Av. Instituto Politénico Nacional, San Pedro Zacatenco, 07360, México, D.F., México
†UiT, The Arctic University of Norway,
Department of electrical engineering (IET),
Lodve Langesgate 2, 8514 Narvik, Noruega

Abstract

This paper presents a solution to the guidance problem, taking into account the geometric properties of the aircraft kinematics. The control design considers a longitudinal dynamics controller that golds the flight path angle and aerodynamic speed at some constant references. The proposed guidance algorithm follows the inner-outer control loops approach. The inner control loop is based on a second-order sliding mode technique, while the outer control loop is designed following a nonlinear geometric control method. Numerical co-simulations employing Matlab-Simulink and the flight simulator X-Plane are presented to verify the performance of the guidance algorithm.

http://www.imavs.org/

1 Introduction

Unmanned aerial vehicles (UAV) applications such as precision agriculture, meteorological monitoring, and ground vehicle tracking require a navigation, flight guidance, and control (NGC) system to track Cartesian plane trajectories. Since fixed-wing aircraft have advantages over rotary-wing vehicles, such as endurance and speed, they can cover a wide surface faster than their counterpart rotary-wing configuration. Thus, fixed-wing aircraft are more suitable for such applications. In fixed-wing aircraft, trajectory tracking in the Crtesian plane requires commanding the lateral-directional dynamic.

The aircraft control literature identifies two approaches to solving fixed-wing aircraft guidance and control problems. In the first approach, the problem is solved with two control loops. An outer loop determines the reference for bank angle based on the desired yaw angle and lateral distance to the desired path. An inner control loop defines the behavior of the control surfaces to follow the references from the outer control loop [1], [2], [3], [4]. In the second approach, a single control loop defines the references and actions of the control surfaces [5]. An intermediate approach integrates guidance and control schemes, as reported in [6].

Reference [2] proposes a conventional proportional and derivative (PD) lateral controller with some nonlinear improvements that enhance UAV tracking performance for different flight conditions, especially recovery from significant cross-track errors. The response of this controller is comparable with the results reported in [3], which proposed a control logic that approximates a proportional-derivative (PD) controller when following a straight-line path and contains anticipatory control terms to enable tracking curved paths. The work in [4] proposes a proportional-integral-derivative (PID) guidance law designed into two parts, a linear time-invariant (LTI) component in the forward path and a time-varying gain in the feedback path; the stability analysis is performed under the circle criterion. The stability properties of the proposed guidance law are less conservative than proportional navigation (PN) [7], proportional-derivative navigation (PDN) [8], and proportional-integral navigation (PIN) [9] guidance laws. In [10], a high-precision, globally stable 3D path tracking algorithm is proposed. The feasibility and performance are evaluated through flight tests. The guidance algorithm combines the smooth representation of the reference trajectory with flatness-based feedforward controls and a nonlinear feedback scheme based on dynamic inversion. Reference [11] develops a guidance law based on "good helmsman" behavior. The integration of the guidance law and the aircraft dynamics, maintaining closed-loop stability, is ensured. An observer to estimate wind data to orient path geometry about the target completes the algorithm. The performance of the algorithm is evaluated using a high-fidelity simulation.

A common characteristic in the works [2], [3] and [4] relies on the geometry of the desired and current aircraft positions and the decomposition of the aircraft lift force when performing a bank turn. A different approach was initiated in [12], where the aircraft kinematics structure is considered to design the guidance algorithm. In [13] the similarities between the aircraft kinematic model and the unicycle mobile robot are reported, pointing out that the only difference is that the aircraft kinematic model

*Email address(es): {marcoa.martinez,hrodriguez}@cinvestav.mx
†Email address(es): jose.j.sanchez@uit.no

cannot change the sign of the aircraft speed, imposing a decisive condition for control design. A control Lyapunov function (CLF) approach is followed to design a guidance algorithm. All feasible constrained inputs are characterized from a CLF for the input constrained case. The feasible set provides the control input.

This paper aims to presents a solution to an aircraft guidance problem using the inner-outer control loops approach. The super-twisting algorithm approach tailors the inner control loop, and the outer control loop is based on nonlinear control methods considering the geometric properties of the aircraft kinematics. In order to verify the performance of the guidance algorithm, a numerical co-simulation is presented using Matlab-Simulink and X-Plane.

This paper has the following structure. Section 2 introduces the aircraft's lateral-directional dynamics. Section 3 is devoted to designing the guidance and control algorithms, while Section 4 reports the numerical simulation results. The paper ends in Section 5 with concluding remarks.

2 Aircraft lateral-directional dynamics

Under the assumption that there is a controller for the aircraft longitudinal dynamics that holds the flight path angle γ and the aircraft aerodynamic speed V at some constant values $\bar{\gamma}$ and $\bar{V}$, respectively, the aircraft lateral-directional dynamics reads as [14][1]

$$\begin{aligned} m\bar{V}\dot{\beta} &= mg(c_\beta s_\phi c_{\bar{\theta}} + s_\beta c_{\bar{\alpha}} s_{\bar{\theta}} - s_{\bar{\alpha}} s_\beta c_{\bar{\theta}} c_\phi) \\ &\quad + m\bar{V}(p s_{\bar{\alpha}} - r c_{\bar{\alpha}}) \\ &\quad + \bar{T} s_\beta c_{\bar{\alpha}} + \bar{q} S C_Y(\beta, p, r, \delta_r) \end{aligned} \tag{1}$$

$$\begin{bmatrix} I_x\dot{p} - I_{xz}\dot{r} \\ I_z\dot{r} - I_{xz}\dot{p} \end{bmatrix} = \begin{bmatrix} -(I_z - I_y)r^2 t_\phi + I_{xz} p r t_\phi \\ -(I_y - I_x) p r t_\phi - I_{xz} r^2 t_\phi \end{bmatrix} + \bar{q} S b \begin{bmatrix} c_l(\beta, p, r, \delta_a, \delta_r) \\ c_n(\beta, p, r, \delta_a, \delta_r) \end{bmatrix} \tag{2}$$

where m is the aircraft mass, β, ϕ, $\bar{\theta}$, $\bar{\alpha}$ are the sideslip, roll, pitch and attack angles, respectively. Moreover, p and r are the roll and yaw angular speeds expressed in body axes, $\bar{T}$ is the power plant thrust, I_x, I_y, I_z and I_{xz} are the aircraft inertia moments, b is the wingspan, S is the aircraft wing surface, and $\bar{q} = 1/2\rho\bar{V}^2$ with ρ the air density. Finally, C_Y, c_l and c_n are the aerodynamic coefficientes of the lateral force and the roll and yaw moments with δ_a and δ_r the ailerons and rudder deflections. In equation (1), $\bar{\theta}$ and $\bar{\alpha}$ are the constant values that agree with $\bar{\gamma} = \bar{\theta} - \bar{\alpha}$.

[1] In the rest of the paper, the notation $\cos(\sigma_1 + \sigma_2) = c_{\sigma_1+\sigma_2}$, $\sin(\sigma_1 + \sigma_2) = s_{\sigma_1+\sigma_2}$ and $\tan(\sigma_1) = t_{\sigma_1}$ for any angles σ_1, σ_2 is considered.

The following equations describe the aircraft lateral-directional kinematic [15]

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} c_1\bar{V} \\ c_2\bar{V} \\ p + t_{\bar{\theta}}\frac{r}{c_\phi} \\ \frac{1}{c_{\bar{\theta}}}\frac{r}{c_\phi} \end{bmatrix} \tag{3}$$

where x and y represent the aircraft Cartesian position, $\chi = \psi - \beta$ is the azimuth angle, and

$$\begin{aligned} c_1 &= -c_{\chi+\beta} s_{\bar{\alpha}+\bar{\gamma}} s_\beta s_\phi c_{\bar{\alpha}} - s_{\chi+\beta} s_\beta c_\phi c_{\bar{\alpha}} \\ &\quad + c_{\bar{\alpha}+\bar{\gamma}} c_{\chi+\beta} c_\beta c_{\bar{\alpha}} + c_{\chi+\beta} s_{\bar{\alpha}+\bar{\gamma}} c_\phi s_{\bar{\alpha}} \\ &\quad - s_{\chi+\beta} s_\phi s_{\bar{\alpha}} \\ c_2 &= -s_{\chi+\beta} s_{\bar{\alpha}+\bar{\gamma}} s_\beta s_\phi c_{\bar{\alpha}} + s_{\chi+\beta} c_{\bar{\alpha}+\bar{\gamma}} c_\beta c_{\bar{\alpha}} \\ &\quad + s_{\chi+\beta} s_{\bar{\alpha}+\bar{\gamma}} c_\phi s_{\bar{\alpha}} + c_{\chi+\beta} s_\beta c_\phi c_{\bar{\alpha}} \\ &\quad + c_{\chi+\beta} s_\phi s_{\bar{\alpha}}) \end{aligned}$$

The lateral control design considers the following assumption on the lateral-directional kinematic and dynamic models.

Assumption 1 The aerodynamic angles are small, this is,

$$\bar{\alpha} \approx 0 \;\rightarrow\; \bar{\gamma} = \bar{\theta}, \quad \beta \approx 0 \;\rightarrow\; \chi = \psi \tag{4}$$

and the lateral force is equal zero, this is,

$$C_Y = 0 \tag{5}$$

The assumption expressed in equation (4) has also been considered in reference [10]. On the other hand, equations (4) and (5) imply that the coordinated turn condition [16]

$$\frac{g}{\bar{V}} s_\phi c_{\bar{\theta}} - r = 0 \tag{6}$$

holds. Finally, the lateral-directional model considered in this work is described by the equation (2) and

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} c_{\bar{\gamma}} c_\chi \bar{V} \\ c_{\bar{\gamma}} s_\chi \bar{V} \\ p + t_{\bar{\theta}}\frac{r}{c_\phi} \\ \frac{1}{c_{\bar{\theta}}}\frac{r}{c_\phi} \end{bmatrix} \tag{7}$$

3 Lateral guidance control

This section presents the control design procedure. First, the control objective is defined as follows.

Control objective Design control inputs δ_a and δ_r such that the lateral aircraft position error $\tilde{y} = y - y_d$ with y_d a constant reference converges to zero, see Figure 1. This work follows the inner-outer control-loop approach to designing the lateral guidance algorithm. The control objective for the inner-loop controller is expressed as follows

$$\lim_{t\to T} \tilde{p} \to 0, \quad \lim_{t\to T} \tilde{r} \to 0 \tag{8}$$

http://www.imavs.org/

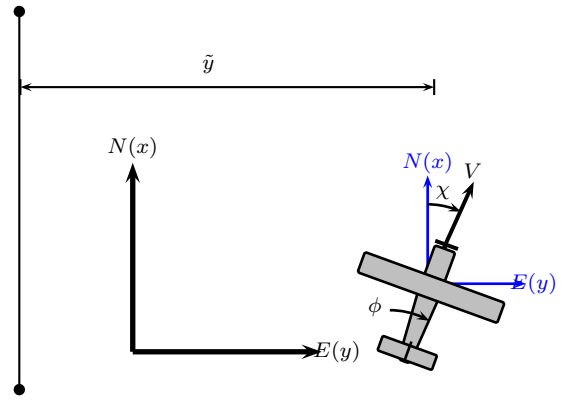


Figure 1: Lateral aircraft error distance $\tilde{y}$.

with T a bounded positive time. Moreover, $\tilde{r} = r - r_d$ and $\tilde{p} = p - p_d$, with p_d and r_d the angular speeds references determined by the outer-loop controller.

A controller with finite-time convergence can be designed as follows. Consider that the aerodynamic moment coefficients have the following structure [15]

$$\begin{aligned} c_l &= c_{l_0} + c_{l_\beta}\beta + \frac{b}{2\bar{V}}c_{l_p}p + \frac{b}{2\bar{V}}c_{l_r}r \\ &\quad + c_{l_{\delta_a}}\delta_a + c_{l_{\delta_r}}\delta_r \\ c_n &= c_{n_0} + c_{n_\beta}\beta + \frac{b}{2\bar{V}}c_{n_p}p + \frac{b}{2\bar{V}}c_{n_r}r \\ &\quad + c_{n_{\delta_a}}\delta_a + c_{n_{\delta_r}}\delta_r \end{aligned}$$

with c_{i_0}, c_{i_β}, c_{i_p} and c_{i_r} with $i = l, n$ the aerodynamic stability coefficients and $c_{i_{\delta_a}}$, $c_{i_{\delta_r}}$ with $i = l, n$ the aerodynamic control coefficients, the rotational dynamics can be expressed as

$$\begin{bmatrix} \dot{p} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \varphi_1(\beta, p, r) \\ \varphi_2(\beta, p, r) \end{bmatrix} + B \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} \tag{9}$$

with

$$B = \bar{q}Sb \begin{bmatrix} I_x & -I_{xz} \\ -I_{xz} & I_z \end{bmatrix}^{-1} \begin{bmatrix} c_{l_{\delta_a}} & c_{l_{\delta_r}} \\ c_{n_{\delta_a}} & c_{n_{\delta_r}} \end{bmatrix}$$

Define

$$\mathbf{S} = \begin{bmatrix} \tilde{p} \\ \tilde{r} \end{bmatrix}$$

then

$$\dot{\mathbf{S}} = \begin{bmatrix} \varphi_1(\beta, p, r) \\ \varphi_2(\beta, p, r) \end{bmatrix} + B \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} - \begin{bmatrix} \dot{p}_d \\ \dot{r}_d \end{bmatrix}$$

The super twisting algorithm proposes the next control input [17][2]

$$B \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} = -\Lambda_2 \int_0^t \mathbf{sign}(\mathbf{S}(\tau))d\tau - \Lambda_1 \mathbf{abs}(\mathbf{S})^{\frac{1}{2}} \mathbf{sign}(\mathbf{S}) \tag{10}$$

where

$$\Lambda_1 = \mathrm{diag}\{\lambda_{11}, \lambda_{12}\},\ \Lambda_2 = \mathrm{diag}\{\lambda_{21}, \lambda_{22}\}$$

with $\lambda_{i1}, \lambda_{i2}$, $i = 1, 2$ positive gains. Thus, one gets

$$\begin{aligned} \dot{\mathbf{S}} &= \begin{bmatrix} \varphi_1(\beta, p, r) \\ \varphi_2(\beta, p, r) \end{bmatrix} - \begin{bmatrix} \dot{p}_d \\ \dot{r}_d \end{bmatrix} \\ &\quad -\Lambda_2 \int_0^t \mathbf{sign}(\mathbf{S}(\tau))d\tau - \Lambda_1 \mathbf{abs}(\mathbf{S})^{\frac{1}{2}} \mathbf{sign}(\mathbf{S}) \end{aligned}$$

the bold functions absolute and sign are vector functions evaluated element-wise.

Consider the following change of coordinates

$$\begin{aligned} \mathbf{S}_1 &= \mathbf{S} \\ \mathbf{S}_2 &= \Theta(\beta, p, r, t) - \Lambda_2 \int_0^t \mathbf{sign}(\mathbf{S}(\tau))d\tau \end{aligned}$$

where

$$\Theta(\beta, p, r, t) = \begin{bmatrix} \varphi_1(\beta, p, r) \\ \varphi_2(\beta, p, r) \end{bmatrix} - \begin{bmatrix} \dot{p}_d \\ \dot{r}_d \end{bmatrix}$$

Then, it follows that

$$\begin{aligned} \dot{\mathbf{S}}_1 &= -\Lambda_1 \mathbf{abs}(\mathbf{S}_1)^{\frac{1}{2}} \mathbf{sign}(\mathbf{S}_1) + \mathbf{S}_2 \\ \dot{\mathbf{S}}_2 &= -\Lambda_2 \mathbf{sign}(\mathbf{S}_1) + \dot{\Theta} \end{aligned} \tag{11}$$

Proposition 1 Assume that matrix B is known, and

$$|\dot{\Theta}| \leq \Theta_0 \tag{12}$$

for some positive bounded Θ_0. Then, there exist positive definite matrices Λ_1 and Λ_2 such that the closed-loop dynamics (9) and (10) satisfy (8).

Proof. Equation (11) can be expanded as follows

$$\begin{aligned} \dot{s}_{11} &= -\lambda_{11}|s_{11}|^{\frac{1}{2}} \mathrm{sign}(s_{11}) + s_{21} \\ \dot{s}_{21} &= -\lambda_{21} \mathrm{sign}(s_{11}) + \dot{\Theta}_1 \\ \dot{s}_{12} &= -\lambda_{12}|s_{12}|^{\frac{1}{2}} \mathrm{sign}(s_{12}) + s_{22} \\ \dot{s}_{22} &= -\lambda_{22} \mathrm{sign}(s_{11}) + \dot{\Theta}_2 \end{aligned}$$

where the following notation is considered $S_1 = \begin{bmatrix} s_{11} & s_{12} \end{bmatrix}^\top$, $S_2 = \begin{bmatrix} s_{21} & s_{22} \end{bmatrix}^\top$ and $\Theta =$

[2] This is, for any vector $\chi \in \mathbb{R}^2$

$$\mathbf{sign}(\chi) = \begin{bmatrix} \mathrm{sign}(\chi_{11}) \\ \mathrm{sign}(\chi_{12}) \end{bmatrix},\ \mathbf{abs}(\chi)^{\frac{1}{2}}\mathbf{sign}(\chi) = \begin{bmatrix} |\chi_{11}|^{\frac{1}{2}}\mathrm{sign}(\chi_{11}) \\ |\chi_{12}|^{\frac{1}{2}}\mathrm{sign}(\chi_{12}) \end{bmatrix}$$

http://www.imavs.org/

$\left[\begin{array}{cc}\Theta_1 & \Theta_2\end{array}\right]^\top$. Thus, (s_{11}, s_{21}) and (s_{12}, s_{22}) define two super-twisting algorithms [17]. As shown in [18], if (12) holds, the states (s_{11}, s_{21}),$(s_{12}, \text{and} s_{22})$ converge to zero in finite-time so that S_1 and S_2 converge to zero in finite time. ◁

Now the control design procedure focus on the kinematic model (7). Expresing (7) in terms of $\tilde{p}$ and $\tilde{r}$ gives

$$\left[\begin{array}{c}\dot{x}\\ \dot{y}\\ \dot{\phi}\\ \dot{\psi}\end{array}\right] = \left[\begin{array}{c}c_{\bar{\theta}}c_{\psi}\bar{V}\\ c_{\bar{\theta}}s_{\psi}\bar{V}\\ p_d + t_{\bar{\theta}}\frac{r_d}{c_\phi}\\ \frac{1}{c_{\bar{\theta}}}\frac{r_d}{c_\phi}\end{array}\right] + \left[\begin{array}{c}0\\ 0\\ \tilde{p} + t_{\bar{\theta}}\frac{\tilde{r}}{c_\phi}\\ \frac{1}{c_{\bar{\theta}}}\frac{\tilde{r}}{c_\phi}\end{array}\right] \tag{13}$$

where the assumption expressed in equation (4) is considered. Assume that $t > T$, and select

$$r_d = \frac{g}{\bar{V}}c_{\bar{\theta}}s_\phi \tag{14}$$

to achieve the coordinated turn condition. As a result, equation (13) becomes

$$\left[\begin{array}{c}\dot{x}\\ \dot{y}\\ \dot{\phi}\\ \dot{\psi}\end{array}\right] = \left[\begin{array}{c}c_{\bar{\theta}}c_{\psi}\bar{V}\\ c_{\bar{\theta}}s_{\psi}\bar{V}\\ p_d + \frac{g}{\bar{V}}s_{\bar{\theta}}t_\phi\\ \frac{g}{\bar{V}}t_\phi\end{array}\right] \tag{15}$$

In equation (15) the only control input is p_d; thus, defining a reference for the roll angle, ϕ the yaw angle ψ can be modified so that the lateral aircraft position y can be driven to the desired value y_d; this is the guidance problem considered in [12]. If the airspeed $\bar{V}$ is not constant, the aircraft translational kinematic is equal to the translational kinematic of the unicycle mobile robot as studied in [13] with the only difference that the speed $\bar{V}$ cannot take negative values in the aircraft case.

As expressed in equation (15), it looks that the configuration space for the yaw angle ψ is $\mathbb{R}$; however, this is not the case; its configuration space is $\mathbb{S}^1$[3], the unit circle. The unit circle can be projected to the Lie group $SO(2)$ as [19][4]

$$\forall\, \psi \in [0,\, 2\pi] \quad \exists \quad R_\psi \in SO(2)$$

where

$$SO(2) = \left\{R_\psi \in \mathbb{R}^{2\times 2} | R_\psi^\top R_\psi = I, \det(R_\psi) = 1\right\}$$

with $I \in \mathbb{R}^{2\times 2}$ the identity matrix. In local coordinates, one has

$$R_\psi = \left[\begin{array}{cc}c_\psi & -s_\psi\\ s_\psi & c_\psi\end{array}\right] \tag{16}$$

[3]
$$\mathbb{S}^1 = \left\{P \in \mathbb{R}^2 | P^\top P = 1\right\}$$

[4]The roll angle ϕ has also a configuration space equal to $\mathbb{S}^1$ but in this work this fact is not considered.

The aircraft kinematic in (15) in terms of the matrix (16) can be expressed as

$$\begin{array}{rcl}\left[\begin{array}{c}\dot{x}\\ \dot{y}\end{array}\right] &=& \left[\begin{array}{cc}c_\psi & -s_\psi\\ s_\psi & c_\psi\end{array}\right]\left[\begin{array}{c}V c_{\bar{\theta}}\\ 0\end{array}\right]\\ \frac{d}{dt}\left(\left[\begin{array}{cc}c_\psi & -s_\psi\\ s_\psi & c_\psi\end{array}\right]\right) &=& \left[\begin{array}{cc}c_\psi & -s_\psi\\ s_\psi & c_\psi\end{array}\right]\left[\begin{array}{cc}0 & -\frac{g}{V}t_\phi\\ \frac{g}{V}t_\phi & 0\end{array}\right]\\ \dot{\phi} &=& \frac{g}{V}s_{\bar{\theta}}t_\phi + p_d\end{array}$$

in compact form

$$\begin{array}{rcl}\dot{X} &=& R_\psi \mathcal{V}\\ \dot{R}_\psi &=& R_\psi \bar{r}^\wedge\\ \dot{\phi} &=& \bar{r}s_{\bar{\theta}} + p_d\end{array} \tag{17}$$

with

$$X = \left[\begin{array}{c}x\\ y\end{array}\right],\ \mathcal{V} = \left[\begin{array}{c}\bar{v}\\ 0\end{array}\right],\ \bar{r}^\wedge = \left[\begin{array}{cc}0 & -\bar{r}\\ \bar{r} & 0\end{array}\right]$$

Moreover,

$$\bar{v} = \bar{V}c_{\bar{\theta}}, \quad \bar{r} = \frac{g}{V}t_\phi \tag{18}$$

and the map $(\cdot)^\wedge : \mathbb{R} \to \mathfrak{so}(2)$ that projects the angular speed $\bar{r}$ to the Lie algebra $\mathfrak{so}(2)$ of $SO(2)$ is characterized as [20]

$$\mathfrak{so}(2) = \left\{\left[\begin{array}{cc}0 & -\bar{r}\\ \bar{r} & 0\end{array}\right], \forall\, \bar{r} \in \mathbb{R}\right\}$$

Hence, $R_\psi \bar{r}^\wedge$ is the tangent space of $SO(2)$ at R_ψ with angular speed $\bar{r}$.

From equation (17) it is clear that the lateral guidance error $\tilde{y}$ can be only commanded though the matrix R_ψ. Hence, the error between R_ψ and R_{ψ_d} is defined as

$$\tilde{R}_\psi = R_{\psi_d}^\top R_\psi \tag{19}$$

with R_{ψ_d} the desired rotation matrix.

Then, it follows that

$$\dot{X} = R_{\psi_d}\tilde{R}_\psi \mathcal{V}$$

Now, the control objective is to command $\tilde{R}_\psi$ to the identity matrix I in such a way that the translational kinematics asymptotically becomes

$$\dot{X} = R_{\psi_d}\mathcal{V} \tag{20}$$

To define the desired rotation matrix R_{ψ_d} consider that equation (20) can be expressed as

$$\begin{array}{rcl}\dot{x} &=& \bar{v}c_{\psi_d}\\ \dot{y} &=& \bar{v}s_{\psi_d}\end{array} \tag{21}$$

In equation (21) the virtual control input is s_{ψ_d} therefore it must be a signal that takes values in the set $[-1,\, 1]$.

http://www.imavs.org/

http://www.imavs.org/

The following virtual control input is selected, to satisfy that $s_{\psi_d} \in [-1, 1]$,

$$s_{\psi_d} = -\tanh(k\tilde{y})$$

with k a positive constant. Hence, the desired rotation matrix becomes

$$R_{\psi_d} = \begin{bmatrix} \sqrt{1-\tanh(k\tilde{y})^2} & \tanh(k\tilde{y}) \\ -\tanh(k\tilde{y}) & \sqrt{1-\tanh(k\tilde{y})^2} \end{bmatrix}$$

Remark 1 It is easy to verify that R_{ψ_d} belongs to $SO(2)$; as a consequence, the matrix error definition in equation (19) ensures that $\tilde{R}_\psi$ also belongs to $SO(2)$.

The rotation matrix error dynamics is described by

$$\dot{\tilde{R}}_\psi = \tilde{R}_\psi \tilde{\bar{r}}^\wedge \tag{22}$$

where

$$\tilde{\bar{r}} = \bar{r} - \bar{r}_d, \quad \bar{r}_d = (R_{\psi_d}^\top \dot{R}_{\psi_d})^\vee$$

with

$$\dot{R}_{\psi_d} = k\bar{v}s_\psi \begin{bmatrix} -r_{11} & 1-\tanh(k\tilde{y})^2 \\ 1-\tanh(k\tilde{y})^2 & -r_{11} \end{bmatrix}$$

$$r_{11} = \sqrt{1-\tanh(k\tilde{y})^2}\tanh(k\tilde{y}),$$

and the map $(\cdot)^\vee : \mathfrak{so}(2) \to \mathbb{R}$ is the inverse map of $(\cdot)^\wedge$ defined as [20]

$$(\bar{r}^\wedge)^\vee = \left(\begin{bmatrix} 0 & -\bar{r} \\ \bar{r} & 0 \end{bmatrix} \right)^\vee = \bar{r}, \ \forall \bar{r}^\wedge \in \mathfrak{so}(2), \ \bar{r} \in \mathbb{R}$$

Finally,

$$\bar{r} = (R_{\psi_d}^\top \dot{R}_{\psi_d})^\vee - k_R P_a(\tilde{R}_\psi)^\vee \tag{23}$$

with $P_a(\tilde{R}_\psi) = \frac{1}{2}(\tilde{R}_\psi - \tilde{R}_\psi^\top)$ and k_R a positive gain. Thus, one has

Proposition 2 Consider the matrix error dynamics in equation (22) in closed-loop with the control law (23). Then, there exist a gain k_R such that the equilibrium point $\tilde{R}_\psi = I$ is almost globally asymptotically stable.[5]

Proof. The closed-loop dynamics (22)-(23) reads as

$$\dot{\tilde{R}}_\psi = -k_R \tilde{R} P_a(\tilde{R}_\psi) \tag{24}$$

The equilibrium points are characterized by the following matrix algebraic equation

$$0 = -k_R \tilde{R}_\psi P_a(\tilde{R}_\psi)$$

It can be proven that the solutions to the above equation are $\tilde{R}_\psi \pm I$. Linearizing the closed-loop dynamic (24) around both equilibrium points it can be verified that the equilibrium point $\tilde{R}_\psi = -I$ is unstable. Consider now the following Lyapunov function [23]

$$\Phi = \frac{1}{2}\operatorname{trace}(I - \tilde{R}_\psi)$$

The time derivative of Φ is

$$\dot{\Phi} = -k_R \left(P_a(\tilde{R}_\psi)^\vee \right)^2$$

thus, the closed-loop trajectories converge to the set [24]

$$\mathcal{D} = \left\{ \tilde{R}_\psi \in SO(2) \,|\, P_a(\tilde{R}_\psi) = 0 \right\}$$

that only contains the matrices $\tilde{R}_\psi = \pm I$, thus, the proof is completed. ◁

Now, defining, from equation (18)

$$\zeta = \frac{g}{\bar{V}} t_\phi - \bar{r}$$

it follows that

$$\dot{\zeta} = \frac{g}{\bar{V}}(1+t_\phi^2)\left(p_d + \frac{g}{\bar{V}} s_{\bar{\theta}} t_\phi \right) - \dot{\bar{r}} \tag{25}$$

thus, one has

Proposition 3 Assume that $\phi \in (-\pi/2, \pi/2)$. Then, the dynamic system in (25) in closed-loop with the control law

$$p_d = -\frac{g}{\bar{V}} s_{\bar{\theta}} t_\phi + \frac{\bar{V}}{g(1+t_\phi^2)}(-K\zeta + \dot{\bar{r}}) \tag{26}$$

where K is a positive constant has a locally exponentially stable equilibrium point at $\zeta = 0$.

Proof. The closed-loop dynamics (25)-(26) is described by

$$\dot{\zeta} = -K\zeta$$

thus, ζ converges exponentially to zero if the constraint $\phi \in (-\pi/2, \pi/2)$ holds. ◁

Remark 2 This last result shows a disadvantage that arises by not considering the configuration space of the states. Even though there is no physical reason to constraint ϕ to the set (A), it must be constrained due to the use of local coordinates.

The main results reads as follows

Proposition 4 Consider the lateral-directional dynamics (7)-(2) in closed-loop with the controller (10)-(26) and the guidance algorithm (23). Then, there exists gains Λ_1, Λ_2, k, k_R and K such that the error distance $\tilde{y}$ locally asymptotically converges to zero.

[5] An equilibrium solution of a dynamical system is said to be almost globally asymptotically stable if it is asymptotically stable with an almost global domain of attraction, i.e., the domain of attraction is the entire state space excluding a set of Lebesgue measure zero [21], [22].

http://www.imavs.org/

Proof. From propositions 1, 2, 3, and 4, no signals can escape to infinity in a finite time. Hence, to achieve that $\tilde{y}$ converges asymptotically to zero, the control gains need to be selected in such a way that the errors $\tilde{r}$ and $\tilde{p}$ converge to zero faster than ζ and $\tilde{R}_\psi$ converging to I. The gains k_R, K need to be selected in such a way that ζ converges to zero faster than $\tilde{R}_\psi$ converges to I. The control authority to drive $\tilde{y}$ to zero is the aircraft yaw angle which cannot take significant values. Thus, the slower gain must be k.

If the errors $\tilde{r}$, $\tilde{p}$, ζ and $\tilde{R}_\psi$ converge to their references, from equation (20), one has

$$\begin{aligned} \dot{x} &= \bar{v}\sqrt{1-\tanh(k\tilde{y}^2)} \\ \dot{y} &= -\bar{v}\tanh(k\tilde{y}) \end{aligned} \tag{27}$$

From the second equation in (27) it can be shown that $\tilde{y}$ converges to zero. As it can be observed in (27) the speed $\dot{x}$ remain bounded and the speed $\bar{v}$ improves the convergence performance. ◁

4 Numerical simulations

In this section, the numerical simulation results are reported. The control and guidance algorithms are computed in Simulink while the X-Plane flight simulator runs the aircraft dynamic model. The communication between Simulink and X-Plane uses the User Datagram Protocol (UDP) [25].

The remote-controlled aircraft known as Telemaster was modeled in Plane Maker, a program bundled with X-Plane that allows design aircraft, see Figure 2. The aircraft's main parameters are $I_x = 11.671\text{kgm}^2$, $I_z = 17.285\text{kgm}^2$, $I_{xz} = -0.024\text{kgm}^2$, $b = 2.386\text{m}$, and $S = 0.858\text{m}^2$. The aerodynamic control coefficients $c_{n_{\delta_a}} = 0.4762$, $c_{n_{\delta_r}} = 1.5888$, $c_{l_{\delta_a}} = 0.8507$, $c_{n_{\delta_a}} = 0.0.0154$ were obtained using the SIDPAC software introduced in [15].



Figure 2: Telemaster modeled in Plane Maker.

A modified version of the longitudinal dynamic controller introduced in [26] is implemented. The aircraft aerodynamic speed and the flight path angle are regulated at $\bar{V} = 20\text{m/s}$, and $\bar{\gamma} = 1\text{deg}$.

The control gain tunning required some experimental tests. After these test it is concluded that the gain Λ_1 modifies the oscillations after p and r have reached p_d and r_d, respectively. The gain Λ_2 adjusts the time it takes to p and r to reach the desired reference. Small values for Λ_1 and λ_2 produce a slow response and large amplitude oscillations in steady-state. On the contrary, significant values for gains λ_1 and Λ_2 produce a fast response and high-frequency, low amplitude oscillations, which in some cases lead to instability.

The gains K and k_R also required several tests; the tunning criteria avoided extensive paths for the roll angle. Finally, the gain k was lowered little by little to reach a reference change of 100m without saturating the aileron and rudder deflection. Large k values demand a more considerable effort to the lateral controller producing large deflections for control surfaces. Small k values allow significant reference changes, but the lateral convergence error will be slow.In conclusion, the gain k must be tuned according to the required mission characteristics. The tunned controller gains are $\Lambda_1 = \text{diag}\{2,3\}$, $\Lambda_2 = \text{diag}\{5,8\}$, $K = 1.5$, $k_R = 1.25$, and $k = 0.004$.

The aircraft takes off from the Atizapan de Zaragoza local airport (MMJC) near Mexico City. The reported air density was 0.9629kg/m^3. During the first 45s, only the longitudinal controller and a PD yaw controller are working. The PD yaw controller tries to keep the aircraft along the runway. The plane has enough altitude at the second 47 to activate the proposed controller and lateral guidance algorithm.

Figure 3 shows the lateral error behavior. The proposed control and guidance algorithm enters into action at 47s with $y(47) = 22200\text{m}$ and $y_d = 22300\text{m}$. As it can be observed, the lateral error converges to zero, the transients at 130s and 210s are due to the reference change, $y_d = 22400\text{m}$ and $y_d = 22300\text{m}$, respectively. The aircraft reaches the first reference; then it moves 100m to the right; after the error converges to zero, it returns to its first reference, 100m to the left. It is important to mention that the jump at second 52, approximately, is due to the fact that the algorithm tries to orientate the aircraft in the direction of the line before it tries to reduce the lateral error. Figure 4 depicts the attitude error which in $SO(2)$ is measured by the term $(P_a(\tilde{R}_\psi))^\vee$. As it can be observed, this attitude error converges to zero after the transient due to lateral position reference changes. Note that the attitude error is dimensionless. However, the gain k_R can be used to obtain the same dimension as $\bar{r}$,$(1/s)$. Figure 5 shows the time history of roll angle. It can be observed that a roll maneuver is not required unless there is a reference change. Figure 6 presents the roll and yaw rate error $\tilde{p}$ and $\tilde{r}$. The proposed controller tries to keep the roll and yaw rate error in zero. Even if it seems

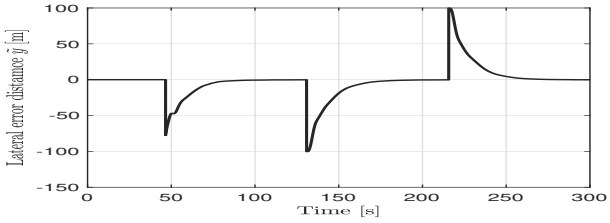


Figure 3: Lateral distance error $\tilde{y}$.

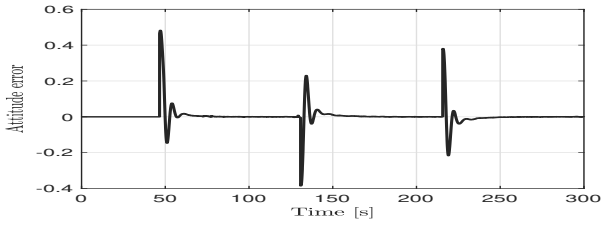


Figure 4: Attitude error $(P_a(\tilde{R}_\psi))^\vee$.

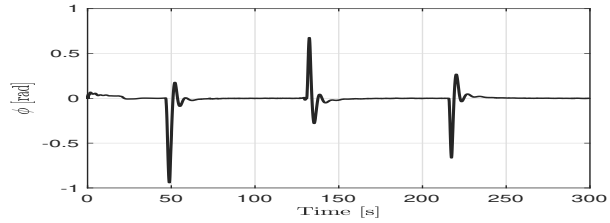


Figure 5: Roll angle.

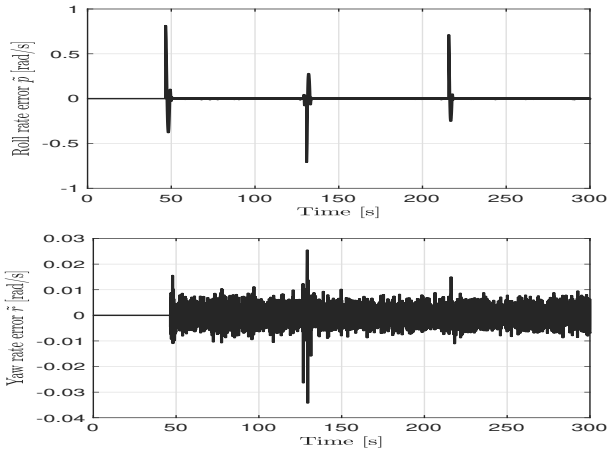


Figure 6: Rotational speed errors p_d (above), r_d (down).

that the yaw rate error is oscillating, these oscillations have a low amplitude, showing that the lateral guidance algorithm is giving good results. Figure 7 shows the control inputs, the aileron, and rudder deflections. Note that the required control surface's deflection remains bounded. Finally, figure 8 presents the coordinated turn condition. As it can be observed, the condition is accomplished with some low amplitude oscillations, except when there is a reference change. The link https://www.youtube.com/watch?v=LXomSgZlFt0 shows the X-Plane simulation.

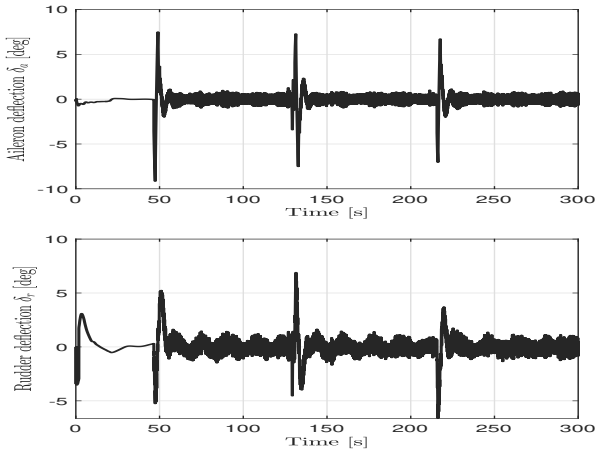


Figure 7: Aileron deflection δ_a (above), rudder deflection δ_r (down).

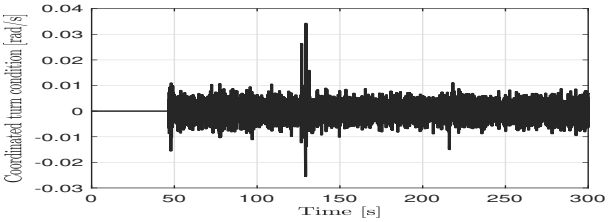


Figure 8: Attitude error $(P_a(\tilde{R}_\psi))^\vee$.

5 Conclusions

A nonlinear controller and a guidance algorithm for the lateral-directional aircraft dynamics were proposed. The nonlinear controller is based on the second-order sliding mode controller, the super twisting algorithm, which guarantees finite-time convergence. The guidance algorithm is designed taking into account the geometric properties of the configuration space of the yaw angle. Numerical simulations showed that the proposed guidance algorithm drives the aircraft lateral error distance

http://www.imavs.org/

to zero. The whole controller and guidance algorithms are implemented in Matlab-Simulink and X-Plane flight simulator.

Acknowledgements

The fitst author thanks CONACYT for the economic support with the scholarships CVU ID: 1079221.

References

[1] Ying Luo, Haiyang Chao, Long Di, and YangQuan Chen. Lateral directional fractional order (pi) α control of a small fixed-wing unmanned aerial vehicles: controller designs and flight tests. IET control theory & applications, 5(18):2156–2167, 2011.

[2] R Samar, Shakil Ahmed, and Faisal Aftab. Lateral control with improved performance for uavs. IFAC Proceedings Volumes, 40(7):37–42, 2007.

[3] Sanghyuk Park, John Deyst, and Jonathan How. A new nonlinear guidance logic for trajectory tracking. In AIAA guidance, navigation, and control conference and exhibit, pages 1–16, 2004.

[4] Mehdi Golestani and Iman Mohammadzaman. Pid guidance law design using short time stability approach. Aerospace Science and Technology, 43:71–76, 2015.

[5] Takeshi Yamasaki, SN Balakrishnan, and Hiroyuki Takano. Integrated guidance and autopilot design for a chasing uav via high-order sliding modes. Journal of the Franklin Institute, 349(2):531–558, 2012.

[6] Syed Ussama Ali, Raza Samar, M. Zamurad Shah, Aamer I. Bhatti, Khalid Munawar, and Ubaid M. Al-Sggaf. Lateral guidance and control of uavs using second-order sliding modes. Aerospace Science and Technology, 49:88–100, 2016.

[7] P Zarchan. Tactical and strategic missile guidance fourth edition. PROGRESS IN ASTRONAUTICS AND AERONAUTICS, 199, 2002.

[8] Pini Gurfil, Mario Jodorkovsky, and Moshe Guelman. Finite time stability approach to proportional navigation systems analysis. Journal of Guidance, Control, and Dynamics, 21(6):853–861, 1998.

[9] Iman Mohammad Zaman and Hamid Reza Momeni. Pi guidance law design using circle criterio. Journal of Control, 4(2):11–19, 2010.

[10] Johannes Stephan, Ole Pfeifle, Stefan Notter, Federico Pinchetti, and Walter Fichter. Precise tracking of extended three-dimensional dubins paths for fixed-wing aircraft. Journal of Guidance, Control, and Dynamics, 43(12):2399–2405, 2020.

[11] Rolf Rysdyk. Unmanned aerial vehicle path following for target observation in wind. Journal of Guidance, Control, and Dynamics, 29(5):1092–1100, 2006.

[12] Marius Niculescu. Lateral track control law for aerosonde uav. In 39th Aerospace Sciences Meeting and Exhibit, pages 1–11, 2001.

[13] Wei Ren and Randy W Beard. Trajectory tracking for unmanned air vehicles with velocity and heading rate constraints. IEEE Transactions on Control Systems Technology, 12(5):706–716, 2004.

[14] José J Corona-Sánchez, Óscar Roberto Guzmán Caso, and H Rodríguez-Cortés. A coordinated turn controller for a fixed-wing aircraft. Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering, 233(5):1728–1740, 2019.

[15] Eugene A Morelli and Vladislav Klein. Aircraft system identification: theory and practice, volume 2. Sunflyte Enterprises Williamsburg, VA, 2016.

[16] Brian L Stevens, Frank L Lewis, and Eric N Johnson. Aircraft control and simulation: dynamics, controls design, and autonomous systems. John Wiley & Sons, 2015.

[17] A. Chalanga, S. Kamal, L. M. Fridman, B. Bandyopadhyay, and J. A. Moreno. Implementation of super-twisting control: Super-twisting and higher order sliding-mode observer-based approaches. IEEE Transactions on Industrial Electronics, 63(6):3677–3685, 2016.

[18] Jaime A. Moreno and Marisol Osorio. Strict lyapunov functions for the super-twisting algorithm. IEEE Transactions on Automatic Control, 57(4):1035–1040, 2012.

[19] J. E. Marsden and T. S. Ratiu. Introduction to Mechanics and Symmetry: A Basic Exposition of Classical Mechanical Systems. Texts in applied mathematics. Springer, 1999.

[20] Marián Fecko. Differential Geometry and Lie Groups for Physicists. Cambridge University Press, 2006.

[21] Taeyoung Lee. Global exponential attitude tracking controls on $\mathsf{so}(3)$. IEEE Transactions on Automatic Control, 60(10):2837–2842, 2015.

[22] E. M. Coates and T. I. Fossen. Geometric reduced-attitude control of fixed-wing uavs. Applied Sciences, 11(7), 2021.

[23] Daniel E. Koditschek. The application of total energy as a Lyapunov function for mechanical control systems. Contemporary Mathematics, 97:131, 1989.

[24] H. K. Khalil. Nonlinear Systems. Pearson Education. Prentice Hall, 2002.

[25] Peter Thomas. X-Plane Blockset. MATLAB Central File Exchange, Retrieved February 25, 2022.

[26] I. Rosario-Gabriel and H. Rodríguez Cortés. Aircraft longitudinal control based on the lanchester's phugoid dynamics model. In 2018 International Conference on Unmanned Aircraft Systems (ICUAS), pages 924–929, 2018.

http://www.imavs.org/

Panel Method Based Guidance for Fixed Wing Micro Aerial Vehicles

Zeynep Bilgin*
Middle East Technical University, Ankara, Turkey
Murat Bronz†
ENAC, École Nationale de l'Aviation Civile, Université de Toulouse, Toulouse, France
Ilkay Yavrucuk‡
Technical University of Munich, Munich, Germany

http://www.imavs.org/

ABSTRACT

In this study, previously proposed panel method based guidance algorithm is tested in a high fidelity simulation environment and in real-life scenarios with a fixed wing mini unmanned aerial vehicle. Panel method is a numerical tool, borrowed from fluid dynamics domain, that can generate collision free, smooth paths in environments with arbitrarily shaped obstacles. It guarantees convergence to global minima with little computational load. Local vector field generated by the panel method is used to demonstrate a simple path-planning and guidance of the fixed wing micro aerial vehicle with hardware experiments. Experimental results suggest that the proposed algorithm can generate collision free paths for a fixed wing aircraft without violating the vehicle's limited flight envelope.

1 INTRODUCTION

In the air transportation industry, fuel cost and expenses associated with manpower are the first and second largest cost components respectively [1]. Therefore, having fully autonomous vehicles is desirable. Without a pilot present to operate the vehicles, path planning and guidance become important issues to be addressed.

The goal of path planning is to determine a collision-free path for a vehicle from an initial point to a target location. The path planning problem is usually mathematically large in size and computationally expensive, often hard to solve even offline. Therefore, methods that provide fast solutions to dynamic trajectory generation for aerial vehicles, while providing obstacle avoidance and collision avoidance are valuable [2].

Path planning for unmanned aerial vehicles (UAV) is well studied in literature. Classical approaches in path planning for autonomous vehicles in literature can be divided into two sub categories: optimization problem based methods and potential field based methods. The first approach is formulating the problem as a large optimization problem with constraints. Different cost functions can be considered to optimize time, distance, energy consumption, number of maneuvers, etc. Constraints may include dynamics of the vehicle and air traffic regulations[3, 4, 5, 6]. Second approach is employing potential functions where repulsive functions prevent collisions and attractive functions guide the vehicle to its destination [7]. Potential field method for path planning is applied in various fields from wheeled ground vehicles to unmanned aerial vehicles [8, 9, 10, 4, 11, 12]. Main advantage of this approach is its simplicity and easy implementation. The potential field method suffers from two major drawbacks. First one, the vehicle may get trapped in a local minima and secondly, the vehicle cannot reach to its destination if there are obstacles nearby due their repulsive force [13, 14]. Harmonic potential fields, a sub category of potential field based approaches, overcome the local minima limitation and produce constrained and well-behaved robot trajectory in static and dynamic environments [15, 16, 17, 18, 19]. An example of harmonic potential field is the potential field of an irrotational flow of an ideal fluid [20, 12]. The stream functions and streamlines can be obtained by using the potential field and then used as vehicle trajectories for solving the problem of navigation with obstacle avoidance. Analytical solutions can be obtained by combining elementary potentials such as uniform flow, sink, source and vortex, can only be used to generate collision free paths around relatively simple geometries, such as cylinders [21, 22]. To compute the velocity field around complex shaped obstacles a numerical tool, namely panel method can be employed. Previously, panel methods were used for robot motion planning and target tracking [23, 24, 25, 2].

More recently, panel method solution is used for addressing path planning problem in urban air mobility [26], where assuming complete knowledge of the city map, path of each vehicle in an air taxi fleet is determined simultaneously using panel method. Panel method based guidance algorithm was previously tested on quad-rotor platform [27, 28]. Quadrotors do not suffer from minimum speed of turn radius limitations. Hence, following the desired path is not a problem. On the other hand, fixed wing aircraft would stall at low speeds or cannot go backwards or sideways etc. There-

*Email address: e188254@metu.edu.tr
†Email address: murat.bronz@enac.fr
‡Email address: ilkay.yavrucuk@tum.de

fore, generating paths that can be followed by a fixed-wing UAV is a challenging task.
In this paper the method proposed in [26] is tested for fixed-wing platform in a realistic simulation environment and hardware experiments are conducted in France. Experiments are held in a local RC airfield runway located in Muret, France. Aim of this study is to test the previously proposed panel method based guidance algorithm [27, 28] with a fixed wing aircraft.

2 GUIDANCE METHODOLOGY

This paper follows the panel method based guidance algorithm that was previously studied[26, 27, 28].

2.1 Panel Method Preliminaries

Panel method is a numerical tool commonly used in fluid mechanics and aerodynamics domain for calculating the velocity field of the irrotational flow of an ideal fluid around arbitrarily shaped objects. Panel method calculations require the surfaces of the obstacles to be divided into discrete elements called panels (hence the name panel method) as shown in Figure 1. Then, each panel is assigned a flow element (such as sources, doublets or vortices) with unknown strength. In this study, prescribed flow elements are selected as point vortices.

Let D be a region in $\Re^2$ containing an ideal fluid and

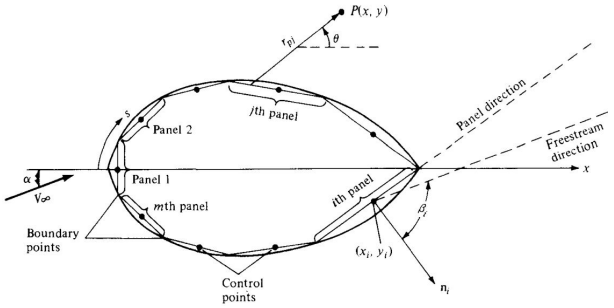


Figure 1: Panel distribution over the surface of a body of arbitrary shape [29].

$\boldsymbol{P} = (x, y)$ be any point in D. Velocity of the fluid can be described as $\boldsymbol{V} = (u, v)$. By Biot-Savart Law, velocity induced by a point vortex element placed at jth panel on any point $\boldsymbol{P} \in D$ can be calculated in terms of unknown vorticity strength γ as shown in Equations 1 and 2.

$$u = \frac{\gamma_j}{2\pi} \frac{(y_j - y)}{r_{pj}^2} \tag{1}$$

$$v = -\frac{\gamma_j}{2\pi} \frac{(x_j - x)}{r_{pj}^2} \tag{2}$$

where $(x_j, y_j)^T$ denotes the position of the point vortex and $r_{pj} = \sqrt{(x - x_j)^2 + (y - y_j)^2}$ is the distance between point vortex element and the point $\boldsymbol{P} = (x, y)^T$. Similarly, velocity induced by jth panel on ith panel can be expressed as $\boldsymbol{V}_{ij} = (u, v)_{ij}$ given in Equation 3.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{\gamma_j}{2\pi r_{ij}^2} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} x_j - x_i \\ y_j - y_i \end{bmatrix} \tag{3}$$

To find the unknown vorticity a boundary condition has to be enforced: the normal component of velocity induced at obstacle surface has to be zero such that no flow can go in or out of the obstacle boundary. After applying this boundary condition, panel method problem can be expressed as a system of linear equations given in Equation 4.

$$K_{ij}\gamma_j = RHS_i \tag{4}$$

$$K_{ij}\gamma_j = (u, v)_{ij} \cdot \vec{n_i} \tag{5}$$

Here, K is a coefficient matrix and γ_j is the unknown vortex strength on panel j. Components of K_{ij} are the normal velocity induced by jth panel on ith panel as given in Equation 5.

$$RHS_i = -u_\infty \cos\beta_i - v_\infty \sin\beta_i \tag{6}$$

The vector RHS consists of normal component of free stream velocity. Solution of the system of linear equations given in Equation 4 gives the unknown vortex strengths on obstacle surface.
When the strength of every flow element in the region $D \in \Re^2$ is known, the fluid velocity at any point can be calculated using Equations 1 and 2 .
Panel method is well studied in fluid mechanics and aerodynamics domain. Reader may refer to [29, 22] for detailed derivation of panel method.

2.2 Panel Method in Guidance Problem

After solving Equation 4 for unknown vortex strengths, fluid streamlines can be obtained. By definition fluid streamlines do not cross each other and avoid collision with objects on their path. Hence, resultant streamlines can be used as vehicle trajectories.
Following [26, 27, 28], panel method equations are further modified to fit the needs of the guidance problem.
First, a point sink element is introduced in RHS vector to represent a goal position for the vehicle (Equation 7).

$$\begin{aligned} RHS_i = &- u_\infty \cos\beta_i - v_\infty \sin\beta_i \\ &- u_{sink} \cos\beta_i - v_{sink} \sin\beta_i \end{aligned} \tag{7}$$

Panel method has built in obstacle avoidance property as streamlines cannot cross obstacle surfaces; however, to avoid collisions between vehicles and additional element is required. To that end, each vehicle is modeled as a point source element and the velocity induced by these sources are included in RHS vector as in Equation 8.

http://www.imavs.org/

$$\begin{aligned} RHS_i = &- u_\infty \cos\beta_i - v_\infty \sin\beta_i \\ &- \sum_{n=1}^{N} u^n_{source} \cos\beta_i - \sum_{n=1}^{N} v^n_{source} \sin\beta_i \\ &- u_{sink} \cos\beta_i - v_{sink} \sin\beta_i \end{aligned} \tag{8}$$

Finally, to increase the safety perimeter around the obstacles *safety source element* introduced in [27] is included in RHS vector in Equation 9.Safety source element is an point source element that travels with the vehicle itself. The safety source element amplifies the vortex strengths on the obstacle edges that are in the vicinity of the vehicle. Consequently, obstacles push the vehicle further away to a safer distance.

$$\begin{aligned} RHS_i = &- u_\infty \cos\beta_i - v_\infty \sin\beta_i \\ &- \sum_{n=1}^{N} u^n_{source} \cos\beta_i - \sum_{n=1}^{N} v^n_{source} \sin\beta_i \\ &- u_{sink} \cos\beta_i - v_{sink} \sin\beta_i \\ &- u_{safety} \cos\beta_i - v_{safety} \sin\beta_i \end{aligned} \tag{9}$$

Here, u_∞ and v_∞ are components of free stream velocity. N is the total number of vehicles. u_{source} and v_{source} are velocities induced by point source elements. u_{sink} and v_{sink} velocity induced by sink element that represents the goal position. u_{safety} and v_{safety} are the velocities induced by *safety source element* introduced in [27].
After all the unknown vortex strengths are found, flow velocity $V = (u, v)$ at any point $P = (x, y) \in D$ can be calculated using the relation given in Equation 10 and Equation 11.

$$u = \frac{-\sigma}{2\pi}\frac{x - x_g}{r_g^2} + \sum_{n=1}^{N} \frac{\sigma_n}{2\pi}\frac{x - x_n}{r_n^2} + \sum_{s=1}^{S}\sum_{k=1}^{K} \frac{\gamma_{sk}}{2\pi}\frac{x - x_{sk}}{r_{sk}^2} \tag{10}$$

$$v = \frac{-\sigma}{2\pi}\frac{y - y_g}{r_g^2} + \sum_{n=1}^{N} \frac{\sigma_n}{2\pi}\frac{y - y_n}{r_n^2} + \sum_{s=1}^{S}\sum_{k=1}^{K} \frac{-\gamma_{sk}}{2\pi}\frac{y - y_{sk}}{r_{sk}^2} \tag{11}$$

Here, goal position is denoted by $P_g = (x_g, y_g)$ and sink strength associated with this goal position is σ. N is the total number of vehicles in the arena and each vehicle has a source assigned to them with strength σ_n. Position of vehicles are denoted as $P_n = (x_n, y_n)$. S and K are number of obstacles and number of panels respectively. Position of a vortex element on a given obstacle and panel is denoted as (x_{sk}, y_{sk}) and associated vortex strength is γ_{sk}. the Euclidean distance between the point P and the goal position is given as r_g. Similarly, r_n is the Euclidean distance between point P and nth vehicle in the region and r_{sk} is the distance between point P and kth panel on obstacle s.

2.3 *Panel Method and Automatic Flight Controller Integration for Guidance*

Block diagram for panel method based guidance is presented in Figure 2. Panel method requires vehicle positions

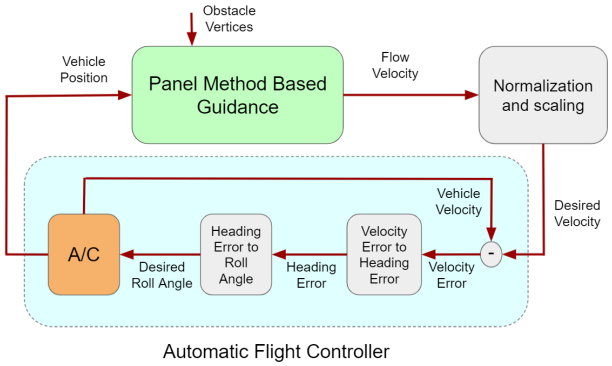


Figure 2: Block diagram for panel method based guidance.

and obstacle position as input. First, Equation 4 has to be solved for unknown vortex strengths. Coefficient matrix K in Equation 4 is a constant and can be calculated and inverted offline to reduce computational load. After vortex strengths are found, flow velocity at current vehicle position can be calculated using Equation 10 and Equation 11. This velocity has to be normalized and scaled. Flow velocity is greater in narrow passages and near goal position; however, the opposite velocity regime is safer and more desirable for vehicles. Hence, flow velocity is also inverted before being fed into automatic flight control system as desired velocity.
Automatic flight controller calculates velocity error and converts it to heading error following the method presented in [30]. Then, heading error is converted to roll angle error constraint by a sustained turn. Finally, desired roll angle is fed to the vehicle.

Reader may refer to [26, 27, 28] for detailed derivation of method.

3 EXAMPLE SCENARIO

In this study, panel method based guidance algorithm is utilized for a fixed wing aircraft flying in an outdoor arena with several no-fly zones. Figure 3 shows an example flight arena. Here horizontal axis is East direction and vertical axis is North direction. All distance units are in meters. Solid magenta shapes represent the no-fly zones and act as obstacles in panel method calculations. One of the strengths of panel method is that it can generate a flow field around arbitrarily shaped objects. Lighter colored regions around no-fly zones are the safety perimeters that are obtained by inflating the object boundaries by an amount of $5m$.

In Figure 4, streamlines and paths followed by different vehicles are plotted. Start position is indicated with a blue circle and goal position is marked with x. At the start position, using the formula given in Equations 10 and 11, desired

http://www.imavs.org/

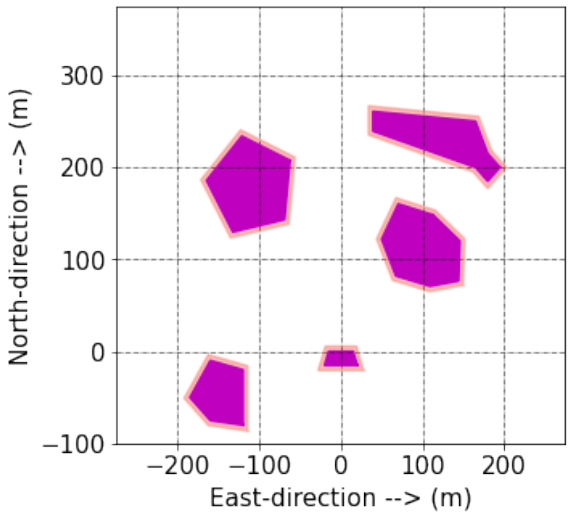
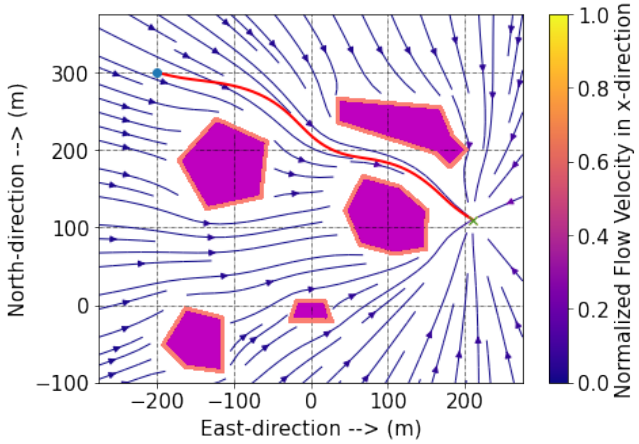
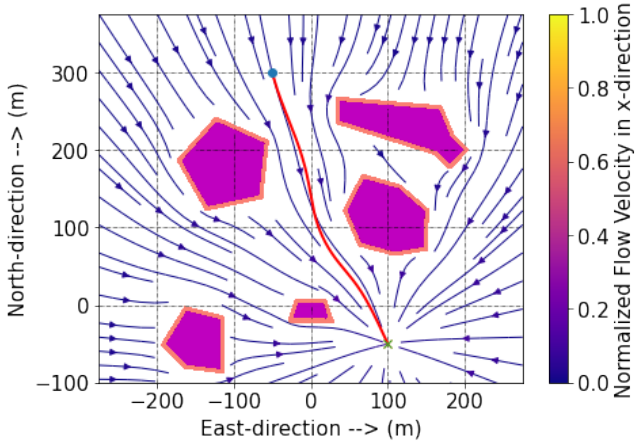


Figure 3: Example flight arena.



(a) Streamlines and path of vehicle 1.



(b) Streamlines and path of vehicle 2.

Figure 4: Streamlines and paths of vehicles with different start positions and destinations.

velocity is calculated. This desired velocity is normalized and fed to the vehicle as a desired heading command. As the vehicle travels in the desired direction, flow velocity calculation is repeated for new position and desired heading command is updated until the vehicle reaches its goal position. Paths plotted in Figure 4a and Figure 4b are the resultant trajectories followed by vehicles from indicated start position to marked goal position. In both cases, vehicles converge to goal position following the streamlines. Since all streamlines lead to the sink placed at the goal position, vehicles would arrive to the destination irrespective of their start position. Furthermore, since goal position can be placed any point on the map, collision free path between any point on the map can be generated.

4 Implementation and Flight Performance

For simulations and real-flight experiments, open-sourced Paparazzi Autopilot system [31] has been employed. The simulation environment resorts on the JSBSIM flight dynamics model and presents a sufficiently realistic demonstration during the development of the panel-method based guidance algorithm. In the following sections, detailed description of the test platform and the flight experiments are discussed.

4.1 Experimental Platform

An overview of the complete flight test system is shown in Figure 5a. A small fixed-wing UAV has been used for the real-flight experiments. A laptop running Unix/Linux for Ground Control Station is used to control the autonomous mission and also to send the reference velocity input calculated by the panel-method based guidance algorithm during flight. An X-Bee radio modem, attached on the side of the computer screen in Figure 5a, is utilized for telemetry between the aircraft and the Ground Control Station and datalink communication was sufficient as the flight tests were done within 500 m radius. An RC-Transmitter is used to ensure the safely recovering of the vehicle in case of loss of control in autonomous mode or any unprecedented situation.

4.2 Flight Experiments

Outdoor runway experiments are conducted in a local RC airfield runway located in Muret, France, that is 124 m $\times$ 12.5 m in size, as shown in Fig. 5b.

ENAC has privileged access to this airfield, and can do tests in a volume of 500 m radius and up to 150 m height (which can be increased to 450 m in certain cases).

For outdoor flight experiments, 2 scenarios are considered. The mission for the first scenario is plotted in Figure 6. Before the mission starts, aircraft is loitering around a standby position just outside the flight mission arena indicated with a start arrow on Figure 6a. Standby position is marked with a red diamond in lower right corner of Figure 6b, and

http://www.imavs.org/



(a) Complete flight test system that is used during the flights.

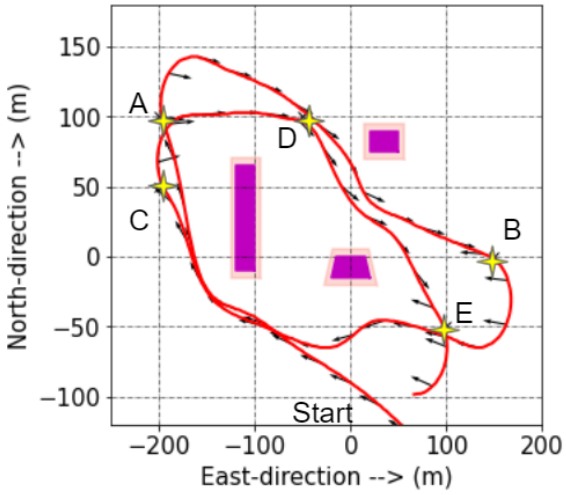


(b) Outdoor flight facility at Muret, France.

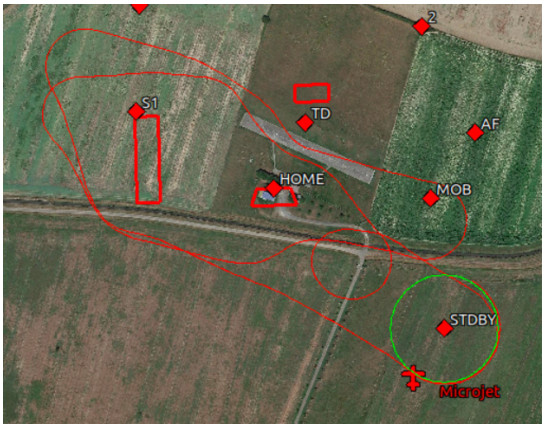
Figure 5: Complete flight test system and flight facility.

Specification		Units
Wing Span	1.2	[m]
Surface Area	0.28	[m^2]
Mass	0.75	[kg]
Battery Capacity	30	[Wh]
Flight speed	12	[m/s]
Flight time	60	[min]
Components		
Motor	T-Motor 2208/18 - Kv 1100	
Autopilot	Paparazzi Chimera v1.0 [1]	
GPS	U-Blox M8N	
Companion board	Raspberry Pi Zero v 1.3	

Table 1: Airframe Specifications



(a) Simulated path and desired velocity vectors.



(b) Simulation for flight facility at Muret, France.

Figure 6: Simulations for Scenario 1.

loiter path is indicated with a green circle around standby point. Goal positions are marked with stars in Figure 6a and no-fly zones are painted with magenta. Furthermore, no-fly zones can also be observed in Figure 6b as red polygons. The mission of the aircraft for the first scenario is to travel between points A-B-C-D-E sequentially without entering the no-fly zones. After reaching point E the vehicle repeats the sequence until it is called back to standby position again. Arrows plotted on the path in Figure 6a are the instantaneous flow velocity calculated by panel method. The difference between the instantaneous flow velocity plotted here and the vehicle velocity is used for heading error calculation in automatic flight controller as discussed in Section 2. For all simulated cases, ground speed of the vehicle was at $13m/s$ and maximum bank angle was limited to 30 degrees.

In Figure 7, path logged in real flight test for scenario 1 in Muret, France is plotted. In this case, the vehicle is to travel between points A-B-C-D-E and repeat the task until it is called back to standby position. Although path followed by the vehicle in real flight is slightly different from the simulated case, the vehicle successfully completes the mission several times without entering into the no-fly zones.

Since all the equations in panel method is non-dimensional, it can be scaled to larger problems. In that respect the second

http://www.imavs.org/

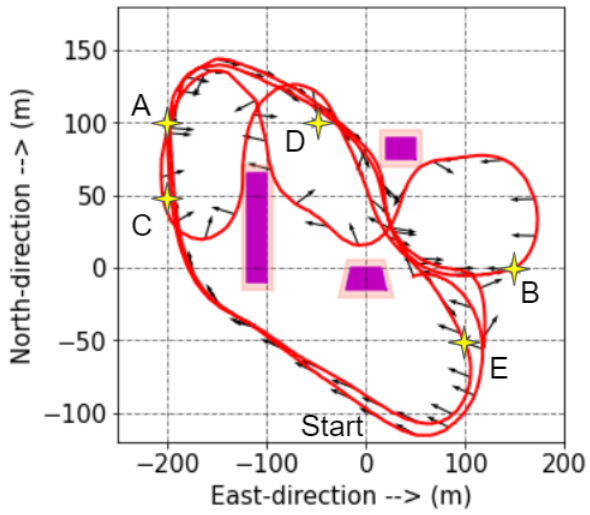


Figure 7: Real flight log for Scenario 1.

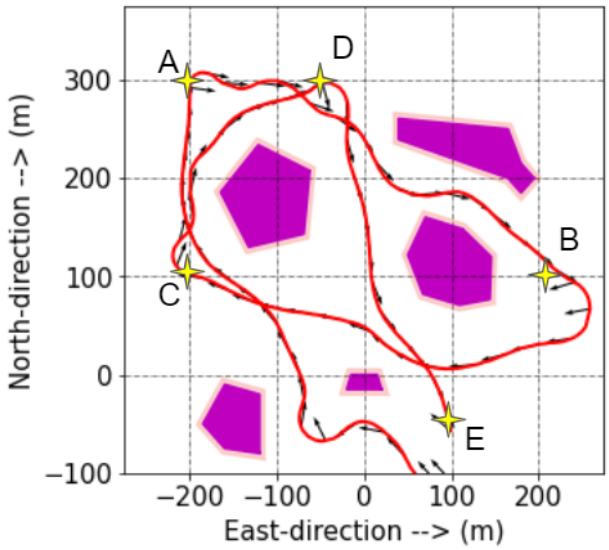


Figure 8: Real flight log for Scenario 2.

scenario is conducted in a larger airspace as shown in Figure 8. Moreover, instead of rectangles, this second arena also contains arbitrarily shaped no-fly zones. Again, the vehicle is loitering around the stand by position outside the map. The mission is to travel between point A-B-C-D-E sequentially and returning to point A again to repeat the task.

In scenario 1, relatively smaller arena is used. When confined in a such narrow space, minimum turn radius of the fixed wing aircraft becomes a limiting factor. Conversely, in scenario 2 much larger area is used; thus, there was enough space for maneuvers at goal locations. To make this scenario more interesting no-fly zones placed to create narrow corridors between goal locations. For real flights, the ground speed of the vehicle was set to $15m/s$ and maximum bank angle was limited to 50 degrees.

5 Conclusion and Future Work

This study focuses on experimental evaluation of the method presented in [27, 28] on fixed wing platform. Guidance for a fixed wing aircraft is a challenging task due to the constraints on minimum speed and turn radius. Panel method imitates fluid flow and generates easy to follow smooth trajectories for vehicles. Simulations and flight experiments indicate that fixed wing aircraft can follow the collision free paths generated by panel method without exceeding the flight envelope limits.

As future work, panel method algorithm can be further improved to generate even smoother paths that can guarantee both obstacle avoidance and envelope protection.

Acknowledgements

The authors would like to thank to Xavier Paris for his support during outdoor flight tests. The author Z. Bilgin was funded by Campus France.

References

[1] M. Ball, C. Barnhart, G. Nemhause, and A. Odoni. Air transportation: irregular operations and control. *Handbooks Oper. Res. Manage. Sci.*, 14:1–67, 2007.

[2] Oguz Uzol, Ilkay Yavrucuk, and Nilay Sezer Uzol. Panel-method-based path planning and collaborative target tracking for swarming micro air vehicles. *Journal of Aircraft*, 47(2).

[3] M.Mirzaei Teshnizi, A. Kosari, S. Goliaei, and S. Shakhesi. Centralized path planning for multi-aircraft in the presence of static and moving obstacles. *International Journal of Engineering*, 33(5).

[4] Yongbo Chen, Guanchen Luo, Yuesong Mei, Jianqiao Yu, and Xiaolong Su. Uav path planning using artificial potential field method updated by optimal control theory. *International Journal of Systems Science*, 2014.

[5] Dantong Ge and Ufuk Topcu. Hierarchical path planning for urban on-demand air mobility. In *2019 IEEE Conference on Control Technology and Applications (CCTA)*, August 19-21 2019.

[6] Priyank Pradeep and Peng Wei. Energy-efficient arrival with rta constraint for multirotor evtol in urban air mobility. *Journal of Aerospace Information Systems*, 16(7).

[7] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 500–505, 1985.

[8] Ulises Orozco-Rosas, Oscar Montiel, and Roberto Sepúlveda. Mobile robot path planning using membrane

http://www.imavs.org/

evolutionary artificial potential field. *Applied Soft Computing Journal*, pages 236–251, April 2019.

[9] Liu Lifen, Shi Ruoxin, Li Shuandao, and Wu Jiang. Path planning for uavs based on improved artificial potential field method through changing the repulsive potential function. In *Proceedings of 2016 IEEE Chinese Guidance, Navigation and Control Conference*, August 12-14 2016.

[10] Z. Yingkun. Flight path planning of agriculture uav based on improved artificial potential field method. In *2018 Chinese Control And Decision Conference (CCDC)*, 2018.

[11] Yuecheng Liu and Yongjia Zhao. A virtual-waypoint based artificial potential field method for uav path planning. In *Proceedings of 2016 IEEE Chinese Guidance, Navigation and Control Conference*, August 12-14 2016.

[12] H. Wang, W. Lyu, P. Yao, X. Liang, and C. Liu. Three-dimensional path planning for unmanned aerial vehicle based on interfered fluid dynamical system. *Chinese Journal of Aeronautics*, 28(1).

[13] Weihao Li, Chenguang Yang, Yiming Jiang, Xiaofeng Liu, and Chun-Yi Su. Motion planning for omnidirectional wheeled mobile robot by potential field method. *Journal of Advanced Transportation*, 2017:1–11, 03 2017.

[14] Seyyed Mohammad Hosseini Rostami, Arun Kumar, Jin Wang, and Xiaozhu Liu. Obstacle avoidance of mobile robots using modified artificial potential field algorithm. *EURASIP Journal on Wireless Communications and Networking*, 2019, 03 2019.

[15] J.-O. Kim and P.K. Khosla. Real-time obstacle avoidance using harmonic potential functions. *IEEE Transactions on Robotics and Automation*, 8(3):338–349, 1992.

[16] Ahmad A. Masoud. A harmonic potential field approach for joint planning and control of a rigid, separable nonholonomic, mobile robot. *Robotics and Autonomous Systems*, 61(6):593–615, 2013.

[17] Farbod Fahimi. *Autonomous robots*. Springer, 2010.

[18] Robert Daily and David M. Bevly. Harmonic potential field path planning for high speed vehicles. In *2008 American Control Conference*, pages 4609–4614, 2008.

[19] Santiago Garrido and Luis Moreno. Robotic navigation using harmonic functions and finite elements. pages 94–103, 01 2006.

[20] X. Liang, H. Wang, D. Li, and C. Liu. Three-dimensional path planning for unmanned aerial vehicles based on fluid flow. In *2014 IEEE Aerospace Conference*, 2014.

[21] R. I. Lewis. *Vortex Element Methods for Fluid Dynamic Analysis of Engineering Systems*. Cambridge Univ. Press, Cambridge, UK, 1991.

[22] J. Katz and A. Plotkin. *Low Speed Aerodynamics*. Cambridge Univ. Press, Cambridge, UK, 2001.

[23] Y. Zhang and K. P. Valavanis. A 3-d potential panel method for robot motion planning. *Robotica*, 15(4).

[24] Jasmin Velagic, Lamija Vuković, and Belma Ibrahimovic. Mobile robot motion framework based on enhanced robust panel method. *International Journal of Control, Automation and Systems*, 18, 11 2019.

[25] Jasmin Velagić, Lamija Vuković, and Belma Ibrahimović. Mobile robot motion framework based on enhanced robust panel method. *International Journal of Control, Automation and Systems*, 18(5):1264–1276, 2020.

[26] Zeynep Unal and Ilkay Yavrucuk. Panel method based path planning for evtol in urban environment. In *Vertical Flight Society Forum 77*, May 10-14 2021.

[27] Zeynep Bilgin, Murat Bronz, and Ilkay Yavrucuk. Experimental evaluation of panel-method-based path planning for evtol in a scaled urban environment. In *Vertical Flight Society Forum 78*, May 10-12 2022.

[28] Zeynep Bilgin, Murat Bronz, and Ilkay Yavrucuk. Experimental evaluation of robustness of panel-method-based path planning for urban air mobility. In *AIAA Aviation Forum*, 27 June-1 July 2022.

[29] John D. Anderson. *Fundamentals of Aerodynamics*. McGraw-Hill, 5th edition, 2011.

[30] Hector Garcia De Marina, Yuri A Kapitanyuk, Murat Bronz, Gautier Hattenberger, and Ming Cao. Guidance algorithm for smooth trajectory tracking of a fixed wing uav flying in wind flows. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 5740–5745. IEEE, 2017.

[31] Gautier Hattenberger, Murat Bronz, and Michel Gorraz. Using the Paparazzi UAV System for Scientific Research. In *International Micro Air Vehicles Conference and Flight Competition*, pages 247–252, Delft, Netherlands, 2014.

http://www.imavs.org/

Reconnaissance Target Grouping and Assignment Considering Fault Severity for UAV Swarm

Taegyun Kim[1], Jingu Kim[2], Ilwon Seo[2] and Seungkeun Kim[1] *
[1] Chungnam National University, Republic of Korea
[2] Poongsan R&D Institute, Republic of Korea

ABSTRACT

This study proposes an optimal task assignment algorithm by using the Hungarian algorithm considering the fault severity and the number of targets. In order to ensure that the efficiency of the swarm reconnaissance mission does not decrease even under the UAV fault, the distance to the targets and the reconnaissance target maximum number are considered as costs. First, we construct the cost calculation formula according to the degree of fault by defining the fault severity. Second, target grouping is performed using the k-means algorithm. Then, to analyze the performance of the proposed algorithm, the simulations are done according to the target number and the ratio of faulty UAVs in the swarm.

http://www.imavs.org/

1 INTRODUCTION

1.1 Background

Unmanned aerial vehicles (UAVs) are employed in various real-life applications, such as surveillance and reconnaissance. Because of their ability to reduce the risk to humans and provide cost-effective options that can be used in several conditions where the operating system cannot be used. With the development of communication technology, multiple UAVs, the swarm flight, can simultaneously perform missions. A group of UAVs can acquire more data and occupy a wider area during surveillance and reconnaissance missions than a single UAV [1]. In addition, the survival rate is higher than when a single UAV is operated on the battlefield. However, even if rotary UAVs have been studied and operated due to their vertical flight capability, they sometimes experience accidents and get a fault. In order to overcome the problem of fault during flight operation, it is important to diagnose the fault and assign and proceed with the mission thereafter.

1.2 Related work

Recently, various studies have been conducted to optimize swarm task assignments by using the Hungarian algorithm [2]–[3]. Amir *et al.* propose a matching algorithm; it matches the UAV to the best nesting station considering UAV's energy and distance to the station [4]. Arezoo *et al.* proposed an algorithm, namely, the Duplicate Agent Hungarian Based Algorithm, for task allocation for UAV package delivery [5]. Moon *et al.* studied and developed an approach to reduce the maximum UAV movement distance to increase the operating time using the Fair Hungarian Algorithm [6]. From the above research cases, it can be seen that the Hungarian Algorithm is widely used for task assignments during swarm missions.

1.3 Main contribution

In this study, the cost calculation is different according to the fault condition by giving the classification of fault severity. Through this, it is possible to perform the optimal task assignment of the swarm. The main contribution in this study are summarized as follows.

The first contribution is the swarm task assignment algorithm considering the fault severity, which can increase the efficiency of mission performance even if the fault occurs during swarm operation.

The second contribution is to minimize each UAV's movement distance and maximize the number of reconnaissance targets to improve swarm operation performance.

Lastly, since the proposed algorithm can calculate only the fault diagnosis result, it can be linked with various fault diagnosis modules.

This paper is organized as follows: Section 2 defines the fault severity and shows the target clustering by the k-means algorithm. Afterwards, the task assignment algorithm constructed based on the Hungarian algorithm will be described. Section 3 compares the performance of the minimization distance algorithm and the proposed algorithm by simulating them in the same environment. Section 4 offers conclusions of the paper with future work.

2 CONSIDERING FAULT SEVERITY FOR UAV SWARM

2.1 Fault severity definition

In performing a flight mission, fault situations occur due to various factors. In this process, there may be cases in which missions can be performed limitedly due to prop damage, etc., or mission continuation is impossible due to a fault of the actuator. Therefore, in order to increase mission efficiency and success rate, it is necessary to assign different missions according to the fault severity. The more severe the fault of the UAV, the lower its mission capability can be. Therefore, by defining the severity of such fault, it is

*Email address: skim78@cnu.ac.kr

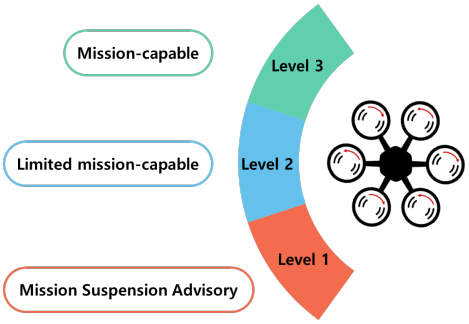


Figure 1: Definition of the fault severity

necessary to classify the task performance. Figure 1 is a definition of the severity levels according to the fault.
In this paper, the UAV severity of the fault is classified into three levels. Level 3 is a mission-capable state, which is non-faulty or close to a healthy state that can carry out the mission. Level 2 is a limited mission-capable state, where failures have occurred but are not fatal to continue missions. Level 1 is a mission suspension advisory state, and a fatal fault occurs, and the UAV must be returned home.

2.2 *Target grouping*

In order to allocate multi targets to available UAVs, clustering is performed according to the number of UAVs. So, the k-means algorithm is used for target clustering. K-means clustering [7] is an iterative, data-partitioning algorithm that assigns n observations to precisely one of the k clusters defined by centroids, where k is chosen before the algorithm starts. In this study, n is the total number of targets, and k is the number of UAVs capable of reconnaissance missions. Fiugre 2 shows a flow chart of the target grouping algorithm.

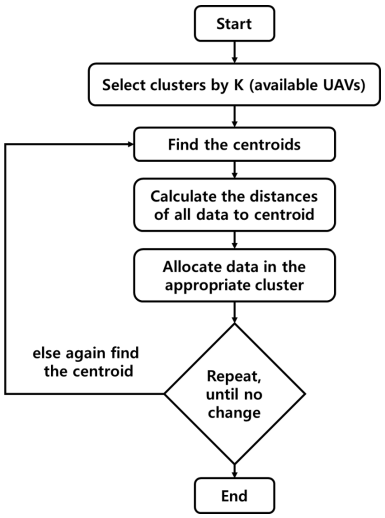


Figure 2: A flow chart of the target grouping algorithm

The center of the target cluster is calculated using the k-means++ algorithm. For $m = 1, ..., n$ and $p = 1, ..., j–1$, select centroid j at random from X with probability k-means++ algorithm is as follows:[8].

$$\frac{d^2(x_m, c_p)}{\sum_{h;x_h \in C_p} d^2(x_m, c_p)} \quad (1)$$

where $d(x_m, c_j)$ denote the distance between c_j and observation m, c_p is the set of all observations closest to centroid c_p and x_m.

2.3 *Task assignment algorithm*

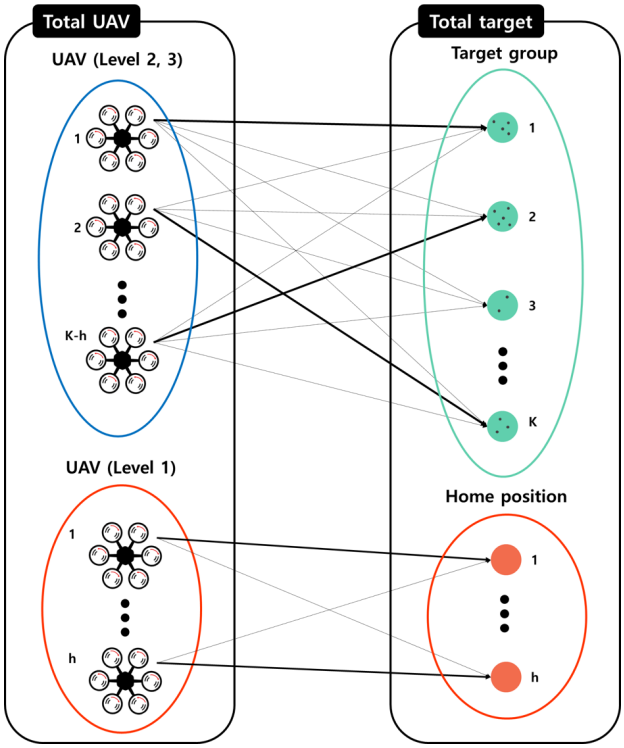


Figure 3: Bipartite graph (Task assignment)

In this paper, we decouple the problem into two sequential optimization subproblems. The first subproblem allocates reconnaissance locations where the center of the target cluster. The second subproblem is to find the optimal task assignment for each UAV using the fault severity level. The assumptions of this paper are as follows:

1) Throughout the allocation process, the attributes of all UAVs and tasks are unchanged.
2) Each UAV knows its own and other UAVs' datas and the number of tasks.
3) All UAVs at each level is controlled through fault tolerance control. However, in the case of level 1, it is assumed that the mission continuation is impossible, and a return to home is necessary.

http://www.imavs.org/

http://www.imavs.org/

Figure 3 illustrate the proposed algorithm for task assignment by a bipartite graph. k is the number of total UAVs and h is the number of level 1 UAVs. The proposed task assignment algorithm is based on a Hungarian algorithm. The Hungarian algorithm is an optimization method for solving assignment problems to maximize or minimize a cost [9]. In this problem, the main goal is to minimize the distance and maximize the target number under successful reconnaissance. The assignment problem is summarized in Algorithm 1.

C is a $k \times k$ cost matrix that can be computed using the attributes of agents and tasks. X is the assignment matrix. $cost$ is the overall cost. w_{i1} is a weight of each level i. $dist$ is the distance between the UAV and the target cluster center, and $dist_{max}$ is the maximum distance from the target cluster center that the UAV can reconnaissance. $dist_h$ means distance from the UAV to home position. $target_n$ is the number of targets in the cluster. The cost calculation is different according to the determined fault diagnosis severity in the proposed algorithm. Level 3 UAVs put more weight on reconnaissance over many targets rather than minimum distances, and Level 2 UAVs put more weight on going minimum distances.

Algorithm 1 Task Assignment by Hungarian Algorithm

procedure $(X, cost)$ = Hungarian(C)
  $0 \leq w_{31} < w_{21} \leq 1$
  if Level 3 **then**
    $C = w_{31}\frac{dist}{dist_{max}} + (1 - w_{31})\frac{1}{target_n}$
  end if
  if Level 2 **then**
    $C = w_{21}\frac{dist}{dist_{max}} + (1 - w_{21})\frac{1}{target_n}$
  end if
  if Level 1 **then**
    $C = dist_h$
  end if
Step 1 : Subtract the smallest cost in each row from all the costs of its row.
Step 2 : Subtract the smallest cost in each column from all the costs of its column.
Step 3 : Draw lines through appropriate rows and columns so that all the zero costs of C are covered and the minimum number of such lines is used.
Step 4 :
  procedure Test for optimality :
    if the minimum number of covering lines is k, an optimal assignment of zeros is possible and the assignment is finished. **then**
    end if
    if the minimum number of covering lines is less than k, an optimal assignment is not yet possible. Proceed to Step 5. **then**
    end if
Step 5: Determine the smallest cost not coverd by any line. Subtract this cost from each uncovered row, and then add it to each covered column. Return to Step 3.
  end procedure
Return X and $cost$
end procedure

3 Numerical Result and Discussion

3.1 Simulation setup

In this scenario, we operate k UAVs and reconnaissance as many as possible out of n targets in clusters. In addition, it aims to efficiently operate the reconnaissance range by adjusting the reconnaissance distance according to the severity of the UAV fault. To have a better result in the target amount and distance of the reconnaissance task assignment problem, Figure 4 shows a mission problem example where several UAVs are in the swarm considering fault severity. The simulation compares the performance of the proposed algorithm with the algorithm that minimizes only the movement distance. The variables used for the simulation are the total number of targets and the rate of faulty UAVs within the swarm. The performance criterion compares the number of targets under successful reconnaissance to the distance each UAV has moved.

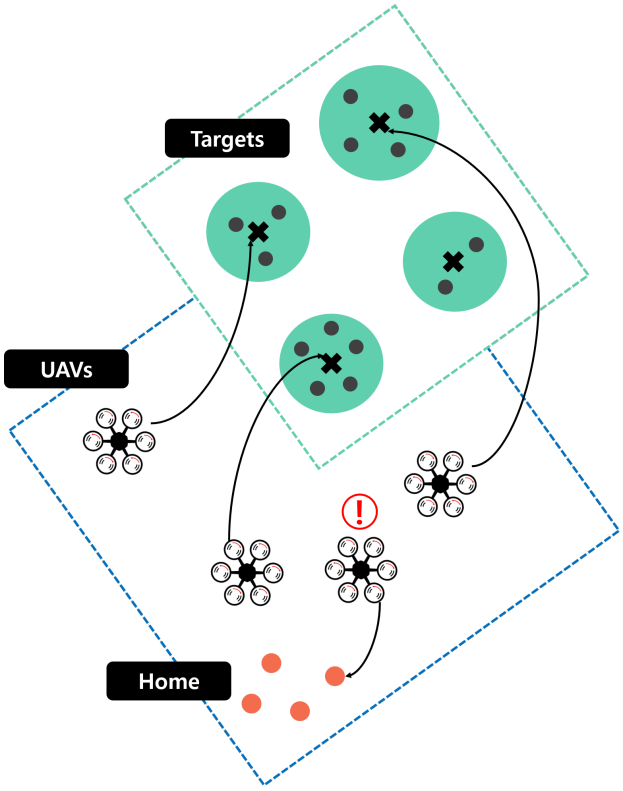
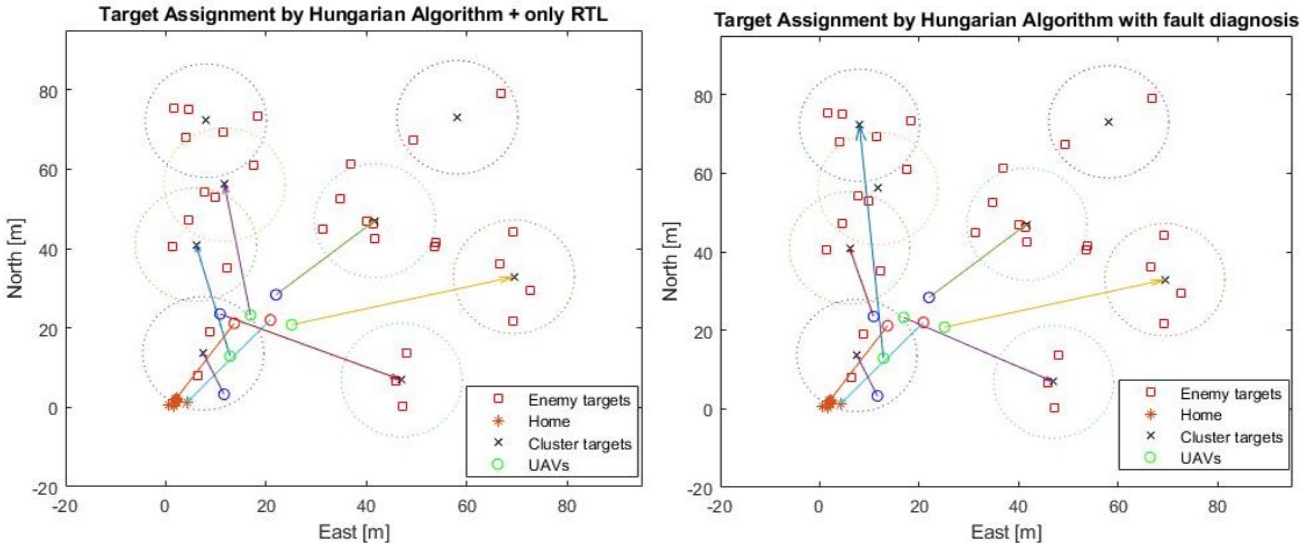


Figure 4: Mission problem description



(a) The case that minimizes only the movement distance

(b) Proposed algorithm

Figure 5: Case 1 simulation

Target number	20		30		50		70	
	Distance minimizing	Proposed algorithm	Distance minimizing	Proposed algorithm	Distance minimizing	Proposed algorithm	Distance minimizing	Proposed algorithm
Search target	14	15	23	25	31	37	48	58
Level 3 UAVs average distance	27.27 m	40.39 m	36.01 m	46.56 m	26.11 m	54.51 m	42.01 m	57.81 m
Level 2 UAVs average distance	31.35 m	21.73 m	26.03 m	18.78 m	24.75 m	17.04 m	30.18 m	22.83 m

Table 1: Comparison of the algorithm performances (case 1)

http://www.imavs.org/

3.2 Simulation results

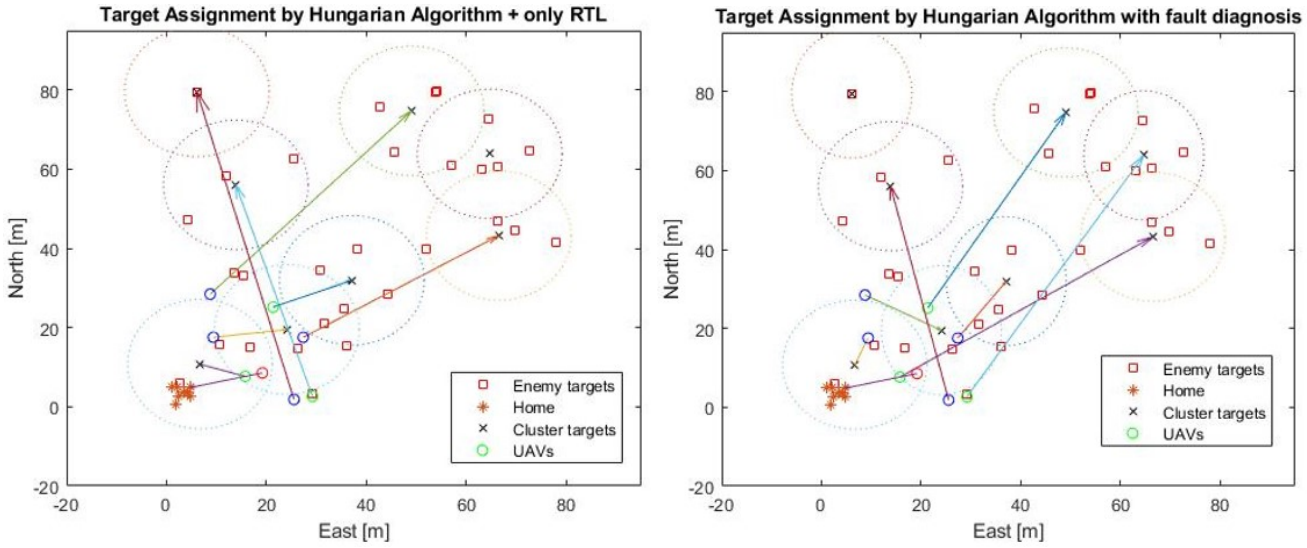
3.2.1 Case 1 Simulation : Total target number

In this scenario, we operate 8 UAVs assuming that the swarm is operated with 3 units of 'Level 3', 3 units of 'Level 2', and 2 units of 'Level 1' to compare the performance of the allocation algorithm while changing the total number of targets. In this simulation, it is assumed that the maximum range that the UAV can reconnaissance is fixed. The simulation is performed when the number of targets is 20, 30, 50, and 70 to compare the performances. Figure 5 (a), (b) shows simulation results when the number of targets is 30 using by minimizing only the movement distance algorithm(a) and proposed algorithm(b), respectively. The green, blue, and red circles are Level 3, Level 2, and Level 1 UAVs.

Referring to Figure 5(a), only the minimum distance between each UAV and cluster target is considered and assigned to the nearest cluster target location. Because the fault severity is not considered, the UAV with the fault is assigned to a distant location. On the other hand, when using the proposed algorithm, as shown in Figure 5(b), the Level 3 UAV goes to a distant place. The Level 2 UAV is assigned to a relatively nearby location. Table 1 compares the average performances of 30 simulations of two cases according to the number of targets. The proposed algorithm assigns the Level 3 UAV to a more distant location on average, and the Level 2 UAV to a nearer location in all simulation cases. Also, more targets could be under successful reconnaissance by considering the number of targets in the cost.

3.2.2 Case 2 Simulation : Rate of fault UAVs

Case 2 simulation compares and analyzes the performance according to the rate of faulty UAVs. We operate 8 UAVs, and the performance in the same environment is compared while changing the ratio of Level 3 and Level 2 UAVs. In Figure 6(a), (b) a simulation results are given when the number of targets is 30. The swarm is operated with 3 units of 'Level 3', 4 units of 'Level 2', and 1 unit of 'Level 1'. When the proposed algorithm is used, the movement distance of the Level 3 UAV increases, but the movement distance of the Level 2 UAV decreases compared to the minimized distance



(a) The case that minimizes only the movement distance (Level 2: 4)

(b) Proposed algorithm (Level 2: 4)

Figure 6: Case 2 simulation

Faulty UAV number	Level 3 : 5 Level 2 : 2 Level 1 : 1		Level 3 : 4 Level 2 : 3 Level 1 : 1		Level 3 : 3 Level 2 : 4 Level 1 : 1		Level 3 : 2 Level 2 : 5 Level 1 : 1	
	Distance minimizing	Proposed algorithm	Distance minimizing	Proposed algorithm	Distance minimizing	Proposed algorithm	Distance minimizing	Proposed algorithm
Search target	27	27	26	28	26	29	26	26
Level 3 UAVs average distance	42.31 m	52.30 m	32.78 m	57.24 m	35.33 m	59.65 m	36.62 m	80.18 m
Level 2 UAVs average distance	43.81 m	13.75 m	29.33 m	24.15 m	49.17 m	30.12 m	40.15 m	30.21 m

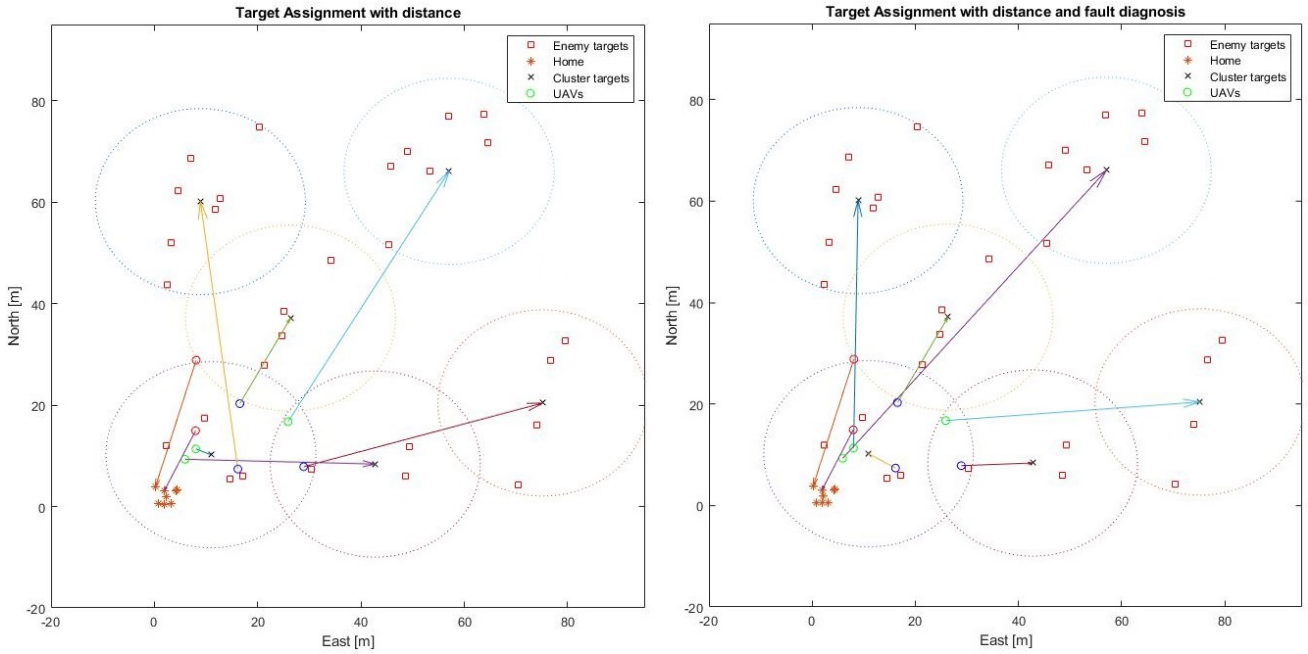
Table 2: Comparison of the algorithm performances (case 2)

algorithm in the same way as in the case 1 simulation. Table 2 compares the average performances of 30 simulations of the distance minimizing algorithm and the proposed algorithm according to the rate of fault UAVs.
In most cases, the proposed algorithm monitered more targets than distance minimizing algorithm. However, if there are much more Level 3 UAVs or even more Level 2 UAVs, they all have the same number of targets found. It is judged that similar results were obtained because the effect was reduced by giving weight according to the level while being driven to one side. Nevertheless, the movement distance of the Level 2 UAV was reduced in all cases. In some cases, the distance minimizing only algorithm can be seen that Level 2 moves more than Level 3. These results mean that the proposed algorithm can operate the faulty UAVs more efficiently.

3.2.3 Case 3 Simulation : All target assignments

In Case 3, simulation compares the performance when the clustering algorithm is performed only with Level 2 and 3 aircraft that can participate. We operate 8 UAVs assuming that the swarm is operated same condition with case 1 scenario. In Figure 7 (a), (b) a simulation results are given when the number of targets is 30. In Figure 7 (a), (b), each UAV is assigned to a close target cluster before considering the severity of faults. However, using the proposed algorithm, it can be confirmed that the Level 3 UAV is a little further away and is assigned to a swarm with many targets.
Table 3 shows the performance comparison of Level 3 UAV and Level 2 UAV according to the total number of targets. 50 simulations were performed with the same scenario, and the average values are summarized in Table 3. Level 3 target means the sum of the number of targets assigned to Level 3 UAV, and Level 2 target means the sum of the number of targets assigned to Level 2 UAV. When the proposed algorithm is applied, fewer targets are assigned to the Level 2 UAV than

http://www.imavs.org/



(a) The case that minimizes only the movement distance

(b) Proposed algorithm

Figure 7: Case 3 simulation

Target number	20		30		50		70	
	Distance minimizing	Proposed algorithm	Distance minimizing	Proposed algorithm	Distance minimizing	Proposed algorithm	Distance minimizing	Proposed algorithm
Level 3 target	10	10	15	19	26	31	37	47
Level 2 target	10	10	15	11	24	19	33	23
Level 3 UAVs average distance	49.24 m	65.16 m	32.78 m	58.19 m	17.30 m	52.14 m	37.04 m	54.29 m
Level 2 UAVs average distance	32.70 m	17.32 m	40.22 m	13.11 m	56.46 m	24.66 m	31.32 m	19.16 m

Table 3: Comparison of the algorithm performances (case 3)

when only the distance is considered, and the Level 2 UAV moving distances are shorter than only distance considered cases.

4 CONCLUSION

This paper proposes a task assignment algorithm for optimal target reconnaissance considering the severity of faults for swarm UAV operations. The fault severity was defined in three levels according to the degree of fault, and the cost calculation was summarized accordingly. The calculated cost was optimized based on the Hungarian algorithm, and the target reconnaissance mission assignment was performed based on this result. Targets were clustered based on the k-means clustering algorithm to maximize reconnaissance targets.
To verify the performance of the proposed algorithm, simulations were performed in the same environment as the algorithm considering only the distance minimization. When using the proposed algorithm, the movement distance of the Level 2 UAV was reduced, and at the same time, more targets could be monitored. As future research, it is planned to perform a swarm flight test after assigning a mission in conjunction with a real-time fault diagnosis module.

ACKNOWLEDGEMENTS

This research was supported by Unmanned Vehicles Core Technology Research and Development Program through the National Research Foundation of Korea(NRF) and Unmanned Vehicle Advanced Research Center(UVARC) funded by the Ministry of Science and ICT, the Republic of Korea(2020M3C1C1A01083162) and funded by the poongsan 'Coaxial rotor type drone for swarm control study'.

http://www.imavs.org/

http://www.imavs.org/

REFERENCES

[1] Taegyun Kim, Seungkeun Kim, and Jinyoung Suk. Formation flight simulation and flight test of multiple ducted-fan uav. *Journal of Institute of Control, Robotics and Systems*, 25:398–406, 2019.

[2] Daniel Hernández, José M Cecília, Carlos T Calafate, Juan-Carlos Cano, and Pietro Manzoni. The kuhn-munkres algorithm for efficient vertical takeoff of uav swarms. In *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, pages 1–5. IEEE, 2021.

[3] Junyan Hu, Hanlin Niu, Joaquin Carrasco, Barry Lennox, and Farshad Arvin. Fault-tolerant cooperative navigation of networked uav swarms for forest fire monitoring. *Aerospace Science and Technology*, 123:107494, 2022.

[4] Amir Mirzaeinia, Savannah Bradley, and Mostafa Hassanalian. Drone-station matching in smart cities through hungarian algorithm: power minimization and management. In *AIAA Propulsion and Energy 2019 Forum*, page 4151, 2019.

[5] Arezoo Samiei, Md Arifin Arif, Trevor Karpinski, Fengyu Wang, and Liang Sun. Energy-aware task allocation for drone package delivery. In *AIAA SCITECH 2022 Forum*, page 2238, 2022.

[6] SungTae Moon, Donghun Lee, Dongoo Lee, Doyoon Kim, and Hyochoong Bang. Energy-efficient swarming flight formation transitions using the improved fair hungarian algorithm. *Sensors*, 21(4):1260, 2021.

[7] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[8] Sergei Vassilvitskii and David Arthur. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, 2006.

[9] SungTae Moon. Development of energy-efficient swarming flight system based on the improved fair hungarian algorithm. 2021.

Evaluation of drag coefficient for a quadrotor model

Gautier Hattenberger, Murat Bronz and Jean-Philippe Condomines
ENAC, Université de Toulouse, France
firstname.lastname@enac.fr

http://www.imavs.org/

ABSTRACT

This paper focuses on the quadrotor drag coefficient model and its estimation from flight tests. Precise assessment of such a model permits the use of a quadrotor as a sensor for wind estimation purposes without the need for additional onboard sensors. Firstly, the drag coefficient has been estimated in a controlled environment via wind generator and motion capture system.Later, the evolution of the coefficient is observed for various mass and fuselage shapes. Finally, an estimation method is proposed, based on the least-squares optimization, that evaluates the drag of the quadrotor directly from outdoor flight data. The latter leads the methodology towards an easier adoption in other researchers' systems without the need for complex and expensive flight testing facilities. The accuracy of the proposed method is presented both in simulation, based on a realistic flight dynamics model, and also for real outdoor flights.

1 INTRODUCTION

Wind field measurement is an important aspect of atmospheric science. Especially, boundary-layer meteorology requires wind speed and direction to study the atmospheric processes close to the ground surface. The ideal would be the use of multiple sensor bases capturing the wind information with high spatial and temporal resolution. However, this solution is practically not feasible from the cost and logistics point of view. The use of UAVs has a big impact on this point, and therefore several methods have already been investigated by researchers. A very common way is to use a fixed-wing aircraft and estimate the wind from the flight track [1] or with the help of additional sensors mounted onto the system, such as a multi-hole probe [2, 3] or even a cheaper solution based on a combination of the simple Pitot tube and wind-vanes [4].

On the other hand, the multi-rotors, known as drones, can offer operational easiness over fixed-wing vehicles as they can take off and land vertically within a confined space and be more compact. However, being an often underactuated system, they have to incline towards the desired direction, which is becoming inappropriate for some of the sensors used in fixed-wing vehicles. Additionally, the flight speed of multi-rotors is typically lower compared to fixed-wing vehicles, lowering the dynamic pressure and making it difficult to use multi-hole probes. Hence, [5] compared the use of four different anemometers on a quadrotor. The study revealed that a thermal anemometer could be used, at the cost of modifications to the UAV structure to place it far enough from the disturbances induced by propellers.

The compactness of the multi-rotors can be used as an advantage, and instead of using additional sensors, the motion and attitude of the vehicle can reveal information about the wind speed and direction [6, 7, 8, 9]. Gonzalez et.al. [7] present different possible models such as static, kinematic, or full dynamic. In addition, a methodology to extract the required parameters is given. They have characterized the propulsion system by using a motor test bench during a wind tunnel experiment, and the drag is extracted from outdoor flights at constant ground speed in steady air. The drag is observed to be proportional to the relative airspeed. Schiano et.al [6] and Marino et.al [10], used experiments with a six-axis force balance, a very precise but fragile and expensive system. Finally, [8] presents a nonlinear observer able to accurately predict the wind components, using only low-cost Inertial Measurement Unit (IMU) and ground speed measurements. The drag force is considered proportional to the rotational speed of the motors, which is almost constant during operation, leading to a constant rotor drag coefficient, similar to [7, 11]. Outdoor flights have been performed in [9, 7] and comparison between ground reference measurements are made, showing the feasibility of wind measurement from quadrotors based on IMU and GPS measurements. Note that estimating the vertical components of the wind requires an accurate modeling of the propulsion system and despite that, have not shown satisfactory results in [12].

In the light of these general principles and authors' previous work [13], the present article is organized as followed. First, the problem modeling focuses on the equations of motion, the hypothesis, and limitations, as well as the experimental airframe. Then, the parameter identification method and evaluation of the mass and fuselage shape effect are presented. Finally, simulation and outdoor flight experiments are described and their results are analyzed.

2 PROBLEM FORMULATION

The objective of this study is to estimate the drag coefficient and evaluate the effect of various physical characteristics, such as mass and shape. This section is presenting the dynamic models and the experimental setup that will be used.

http://www.imavs.org/

2.1 Kinematic and aerodynamic model

The kinematic model is the same than a previous work [13] and is very common in the literature [7]:

$$\dot{\mathbf{X}} = \mathbf{V_k} = \mathbf{V_r} + \mathbf{V_w} \tag{1}$$

$$m\dot{\mathbf{V}}_\mathbf{k} = m\mathbf{g} + \mathbf{D}(V_a) + \mathbf{T} \tag{2}$$

where:

- $\mathbf{X}$ is the position vector relative to earth frame
- $\mathbf{V_k}$ is the ground speed vector relative to earth frame (inertial velocity)
- $\mathbf{V_r}$ is the relative air speed vector
- $\mathbf{V_w}$ is the wind speed vector relative to earth frame
- equation 1 represents the wind triangle
- m is the mass of the model and $\mathbf{g}$ the gravity vector
- $V_a = \|\mathbf{V_r}\|$ is the norm of the airspeed
- $\mathbf{D}$ is the drag vector in earth frame, as a function of airspeed
- $\mathbf{T}$ is the control forces vector (thrust) in earth frame

The wind speed is supposed to be constant or slowly varying, therefore the derivative of the wind triangle (equation 1) gives:

$$\dot{\mathbf{V}}_\mathbf{w} = 0 \quad \Rightarrow \quad \dot{\mathbf{V}}_\mathbf{k} = \dot{\mathbf{V}}_\mathbf{r} \tag{3}$$

The drag modeled as a linear function of the relative airspeed:

$$\|\mathbf{D}\| = k\, V_a \tag{4}$$

where k is a constant factor determined with a calibration method or estimated from flight data. The drag is usually quadratic with the airspeed, however the influence of the rotor drag at low speed results in a nearly linear drag [13, 12, 11]. The validity of this model is also discussed in section 3.2.

The control force vector $\mathbf{T}$ can be expressed from the norm of the thrust T_{total} and the orientation of the body relative to earth frame represented by the rotation matrix $\mathbf{R_{0b}}$. This matrix can be computed from Euler angles ϕ, θ and ψ with the classic DCM matrix as in [7].

$$\mathbf{T} = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix} = \mathbf{R_{0b}} \begin{pmatrix} 0 \\ 0 \\ T_{total} \end{pmatrix} \tag{5}$$

In practice, the total thrust is computed from the assumption that the vertical acceleration is null and that the projection of the thrust on the vertical axis is equal to the mass of the drone, resulting in the following formula:

$$T_{total} = \frac{mg}{\cos\phi \cos\theta} \tag{6}$$

2.2 Experimental setup

The custom quadrotor frame presented in [13] have been reused for this experiment. It is a simple cross shape made of thin aluminum bars to hold the motors and it is possible to place a spherical 3D-printed central body around the electronic components and the battery. In the previous work, the spherical body was always used. Since the goal of this new work is to study the influence of mass and shape, several experiments are conducted with different configurations. Starting from the the base configuration, additional masses can be added to increase the weight with changing the overall shape (thus, not changing noticeably the friction drag) or by adding the spherical body over the central part. Since the maximum additional mass that can be added is almost the same than the frame with the sphere, we consider that it allows to compare the quadrotor with the same mass condition but with a different shape. The Figure 1 is presenting the different elements: base frame, extra masses and spherical body.

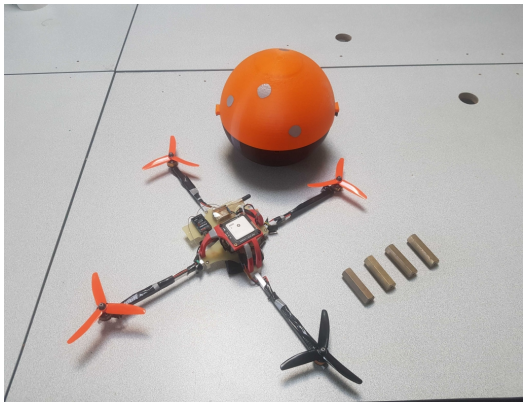


Figure 1: Quadrotor with additional masses and custom spherical body.

The general characteristics and components are summarized in the Table 1. The autopilot software used is the *Paparazzi* UAV System [14]. For indoor calibration and experiments, the localization is provided by a motion capture system from *Optitrack*. For outdoor flights, a GPS receiver is used.

3 EVALUATION OF MASS AND SHAPE

3.1 Parameter identification procedure

The drag coefficient identification methodology is the same than [13], which is a previous work with the same facility. The general idea is to hover with the frame to characterize in front of a wind generator or wind tunnel. In this case, the wind is coming from a *WindShape* wind generator and the indoor position is provided by an *Optitrack* motion capture system. The Figure 2 shows the quadrotor during a calibration.

When the wind is blowing on a frame, the generated drag is balanced by the horizontal thrust coming from the bank

component	characteristic
material	aluminum & plastic (PLA)
motors	T-motor F30
propellers	Dalprop 5x4 (3 blades)
battery	3S, 2200 mAh
autopilot	Tawaki v1 with Paparazzi
GPS	U-blox M8
size (motor to motor)	47 cm
base mass	547 grams
additional masses	up to 520 grams
sphere diameter	22 cm
sphere mass	350 grams
flight time	7 to 15 minutes

Table 1: Quadrotor characteristics

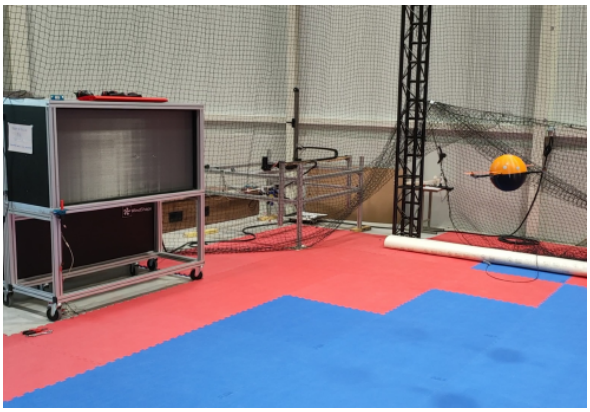


Figure 2: Quadrotor during calibration in front of the Wind-Shape wind generator.

angle. This angle is increasing with the wind speed. In order to mitigate errors due to attitude estimation offset, the frame is turning around its yaw axis. The result is that the roll and pitch angles, estimated by the Attitude and Heading Reference System (AHRS) from inertial measurements, are describing sinusoidal shapes. The Figure 3 is showing the roll angle during the calibration of the quadrotor. This flight correspond to four different wind speeds, with two complete turns of the frame at each speed. The red sine curves are fitted to the telemetry data. The magnitude of each oscillation corresponds to the bank angle of the quadrotor in the direction of the wind generator.

The conducted measurements are as follow:

- Five different quadrotor masses are tested: 0.547 kg (no additional mass), 0.677 kg, 0.807 kg, 0.937 kg and 1.067 kg
- For each mass, estimate bank angle at 9 different wind speeds from 2.5 m/s to 12.5 m/s
- Plot the tangent of the bank angle as function of speed for the different masses.

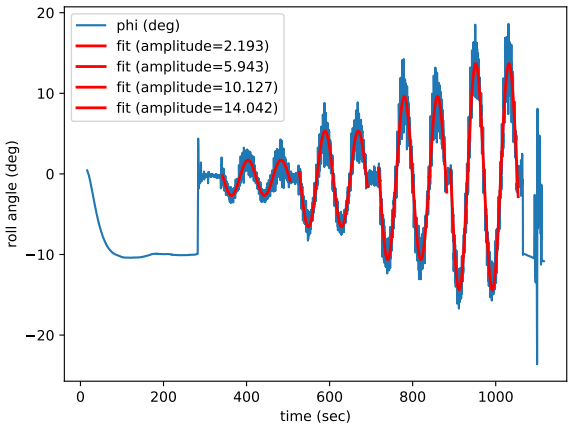


Figure 3: Estimation of the bank angle from sine curve fitting of the roll angle at different wind speed.

The result is shown in Figure 4. The next sections are analyzing the two most important outcomes of these data: the hypothesis of linearity of the drag model and the effect of the mass (and shape) on the drag coefficient.

3.2 Discussion on drag model

The Figure 4 is showing the mean bank angle as a function of wind speed for different mass configurations. It can be observed that between 2 m/s and 9 m/s, the relation between the angle and speed, thus between drag and speed, is linear. After that point, measurements are becoming unreliable due to important perturbations on the frame.

In [12], a similar result is found for a standard DJI quadrotor. At a bank angle of 6 deg, the drag switch from a linear to quadratic evolution, which correspond to an airspeed around 8 m/s to 10 m/s. Standard flight speed for such frame is around 5 m/s. In order to keep the model simple, we will assume that the relative airspeed on the frame stays lower to the critical speed, and only the linear part is considered.

The reference speed for each experiment have been established from the averaged measurements of an hot-wire anemometer. The comparison with other means of measurements (ultrasonic wind sensor) have shown that our reference probe might over estimate by 0.1 m/s to 0.5 m/s the speed. If corrected, the linear model holds down to a null speed.

3.3 Effect of mass and shape on drag

A first observation from Figure 4 is that at a given speed, the bank angle is decreasing with the added weight. This is also depicted in Figure 5. When the weight is increasing, the total thrust is increasing to compensate. The total drag is also increasing, however the bank angle is decreasing. This means that the drag is not increasing as fast as the weight.

Also from the data presented in Figure 4, linear regressions are applied on data lower than 9 m/s, for each of the

http://www.imavs.org/

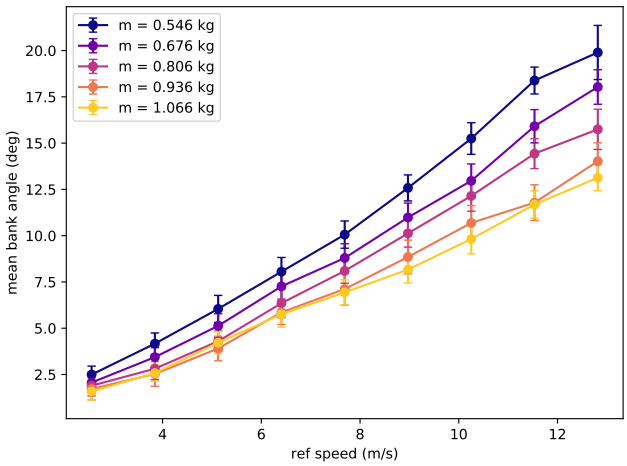


Figure 4: Mean bank angle at different wind speed and different masses; error bars are standard deviations on measurements

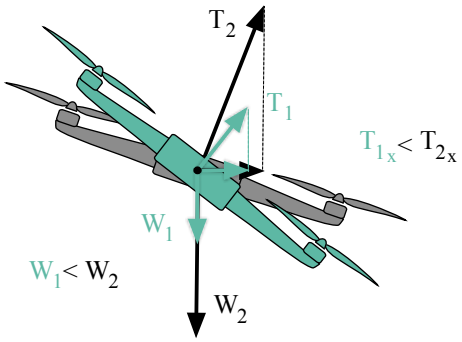


Figure 5: Thrust and weight forces applied on a quadrotor with two different mass. When the mass increases, the thrust increases to compensate the weight, the drag is also increased due to the increased motor speed, however, overall bank angle is reduced.

masses. Then the drag coefficient is computed as in [13]. The Figure 6 is showing the results by plotting the drag coefficients as a function of the mass of the quadrotor. Since the additional weight is not changing the overall shape, the variations of the drag is only related to the variations of motor speed, since a higher thrust is required with increasing weight. A linear regression is applied on the data, showing that the drag coefficient is increasing with the mass.

The influence of the shape is evaluated by comparing the drag coefficient of the quadrotor with and without the spherical body, at the same total weight. The green cross on Figure 6 (top right) is showing the drag coefficient with the sphere, coming from a previous calibration. It can be observed that the coefficient is much bigger, almost doubled by the presence of the large sphere at the center, which is an ex-

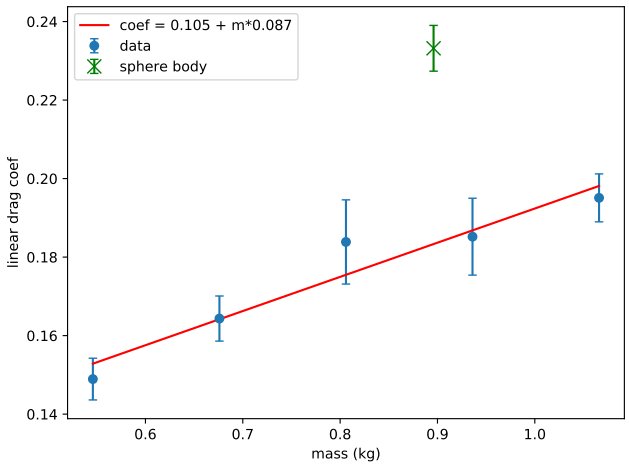


Figure 6: Linear drag coefficient as a function of mass; green cross indicates the model with spherical body; error bars are linear regression standard errors on linear regressions

pected result.

Finally, the Figure 7 is showing the drag coefficient over mass ratio ($\frac{k}{m}$) as a function of the mass. This parameter is related to the sensitivity of the quadrotor to the wind. With higher ratio, the bank angle in hover is increasing at a given speed. As expected from previous curves, it is interesting to lower the mass while increasing the drag. In our case, the lightest frame without the sphere body is giving the best ratio (with the longest flight time as thrust is lower), even compared to the frame with the sphere body, generating more drag at the cost of an extra weight.

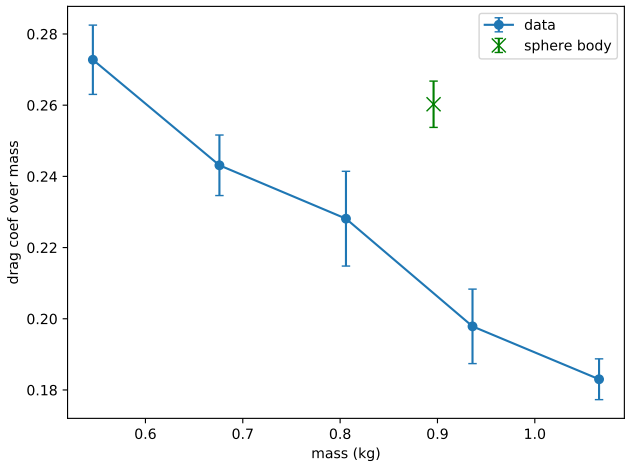


Figure 7: Linear drag coefficient over ratio as a function of mass; Green cross indicates the model with spherical body; error bars are linear regression standard errors on linear regressions

http://www.imavs.org/

http://www.imavs.org/

4 ESTIMATION OF DRAG FROM FLIGHT DATA

The previous section have shown that the weight and shape of the airframe have an important impact on the the drag coefficient. If it is possible to measure this value with a complex experimental setup involving wind tunnels, motion capture systems or forces and torque sensors, it would be more convenient to extract the relevant parameter from flight data directly. The next sections are presenting an approach to estimate the drag over mass $\frac{k}{m}$ ratio from trajectory analysis using a least-square optimization method, followed by simulation and experimental results.

4.1 Estimation using least-square optimization method

Least squares problems are optimization problems expressed in the form

$$\min_{x\in\mathbb{R}^n}\frac{1}{2}\|g(x)\|^2=\frac{1}{2}g(x)^T g(x)=\frac{1}{2}\sum_{i=1}^{m}g_i(x)^2,$$

where $g:\mathbb{R}^n\to\mathbb{R}^m$ is a differentiable function. In particular, these optimization problems arise when one wants to calibrate parameters of a mathematical model by using data. In our application, we are interested in analyzing the relationship between the drag coefficient thought k such that $D=kV$ and the mass m of the quadrotor. It is postulated in [7] that, a simple dynamic particle model named "residual model" can be established in the following way :

$$f_\varepsilon(k/m,\mathbf{W}) = \dot{\mathbf{V}}_\mathbf{k}-(\mathbf{T}_{spec}+\mathbf{D}_{spec}) \quad (7)$$

$$= \dot{\mathbf{V}}_\mathbf{k}-\left(\frac{m\mathbf{g}\bar{\Theta}}{m}-\frac{k}{m}\mathbf{V_a}\right) \quad (8)$$

$$= \dot{\mathbf{V}}_\mathbf{k}-\left(\mathbf{g}\bar{\Theta}-\frac{k}{m}(\mathbf{V_k}-\mathbf{W})\right) \quad (9)$$

where $\mathbf{T}_{spec}$ and $\mathbf{D}_{spec}$ are specific thrust and drag respectively, wind speed is $\mathbf{W}=[W_x,W_y]^T$ and $\bar{\Theta}=\tan(\Theta)$. Thus, the specific coefficient (k/m) and $\mathbf{W}$ are unknown parameters to be determined. The least squares estimation of the unknown parameters is performed by solving the following optimization problem:

$$\min_{k/m,\mathbf{W}}\sum_{i=1}^{m}(\|f_\varepsilon(k/m,\mathbf{W})\|)^2.$$

To find an approximation of a local minimum of the least squares problem, the *Scipy.optimize* package have been used. Several solvers are available for the `least_squares` interface. The two methods tested, *Trust Region Reflective* and *Levenberg-Marquardt* algorithms, are performing with similar results and resolution time.

4.2 Simulation results

In order to evaluate the feasibility of the approach, it is first applied to simulation data. The simulator used is provided by the Paparazzi system and is based on the *JSBSim* flight dynamic model. It can be considered as a high fidelity model that can take into account complex aerodynamic parameters and wind inputs. During the simulation, the mass is fixed to 0.897 kg and the drag coefficient to 0.230.

The trajectory of the quadrotor is a square with a side of 40 meters. Several simulated flights have been recorded at different wind speeds: 0 m/s, 4 m/s and 8 m/s, always coming from the North in NED frame. The commanded ground speed of quadrotor is 2 m/s. Similar results are obtained at 5 m/s, but are not presented here.

The Figure 8 presents the results of the simulation along the X axis (towards North), at the three different wind speeds. The data plotted correspond to the elements of the dynamic equation 7: acceleration, specific drag and specific thrust. The wind is correctly estimated in all cases with an error lower than 0.5 m/s in the worst case. The error (sum of the three aforementioned elements) is always close to zero. It is important to note that the data presented here are based on the perfect simulated values without sensor (IMU, GPS, barometer) noise nor bias.

The Table 2 is the synthesis of the simulation flights, presenting the estimated parameters compared to the values given as input to the simulator. It can be concluded that the method is functional and can provide excellent results on synthetic data.

#	mass sim (kg)	k sim	k/m sim	k/m estimated	W_x est. (m/s)	W_y est. (m/s)
1	0.897	0.23	0.256	0.257	-0.023	0.001
2	0.897	0.23	0.256	0.250	-4.13	0.039
3	0.897	0.23	0.256	0.245	-8.43	-0.006

Table 2: Simulation results for drag over mass coefficient and wind components for three different configurations: 1 - no wind, 2 - 4 m/s wind from North, 3 - 8 m/s wind from North. The commanded ground speed is 2 m/s. The drag and mass correspond to the quadrotor with sphere body. Corresponding acceleration and forces are shown on Figure 8

.

4.3 Experimental flight results

In order to evaluate the method in realistic conditions, the same type of flight than in simulation have been performed with the real frame outdoor. These experimental flights were performed on a small airfield near Toulouse, using a standard GPS for positioning. The Figure 9 is the 3D trajectory of the quadrotor, in full autonomous navigation, at altitudes between 20 and 40 meters above ground.

On the day of the experiment, the average wind reported by public weather stations nearby was between 9 km/h to 11 km/h (2.5 m/s to 3 m/s) from the North (with a variation of $\pm 20°$). Unfortunately, the wind gust was also rather strong, from 5 m/s to 7 m/s.

http://www.imavs.org/

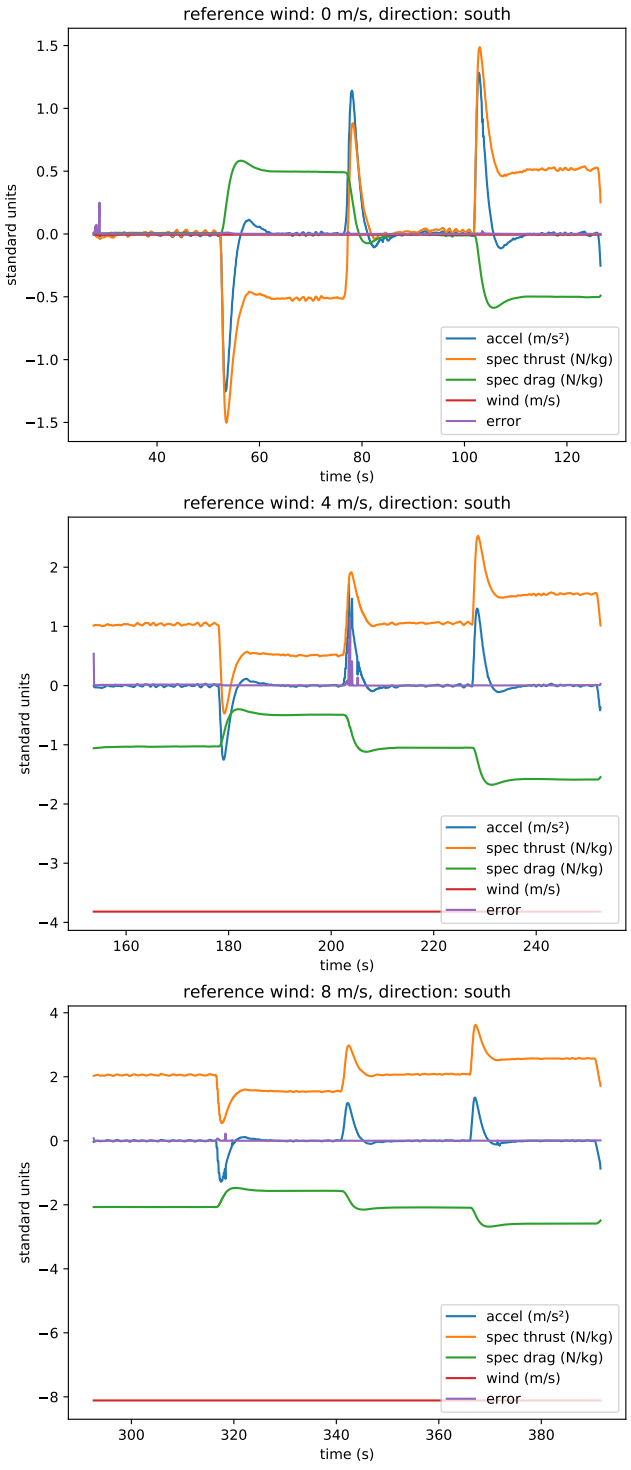


Figure 8: Evolution of acceleration, specific drag, specific thrust and equilibrium error during a square trajectory in simulation. Results are presented for the X axis only, at a wind speed of 0 m/s (top), 4 m/s (middle) and 8 m/s (bottom)



Figure 9: Square trajectory for outdoor experiment. The four segments are oriented North/South or West/East

Three different configuration of the quadrotor have been tested:

1. the light frame, with no additional mass
2. the heavy frame, with all the additional mass but with the same general shape than the light frame
3. the spherical body is placed around the electronics, resulting in a different shape but with almost the same weight than the heavy configuration

On the flight with the light configuration is detailed on Figure 10. Since the raw flight data are very noisy, they need to be filtered before applying the least-square optimization in order to obtain meaningful results. A *Savitzky-Golay* filter from the *scipy.signal* package is applied to all the input signals (ground speed and attitude angles) with the same parameters to avoid phase-shift issues. The acceleration is also obtained with this filter used as a differentiation filter on the ground speed. The first remark is on the wind estimate that seems to be very far from the expected value, in particular with a strong component towards the East. As mentioned before, the wind was rather turbulent on the day of experiment, and perceived direction and intensity by the operator (rather strong from NW) where actually closer to the estimated values than weather data.

The Figure 11 is specifically showing the residual error along the X and Y axis. This error is the sum of the acceleration minus the specific thrust and drag forces. As we can see, the fitting is rather good during the legs and constant speed, but the blue and orange curves are not matching well during the strong acceleration phases, corresponding to flight direction changes at each corner. A part of the problem might also come from the Savitzky-Golay filter smoothing the data.

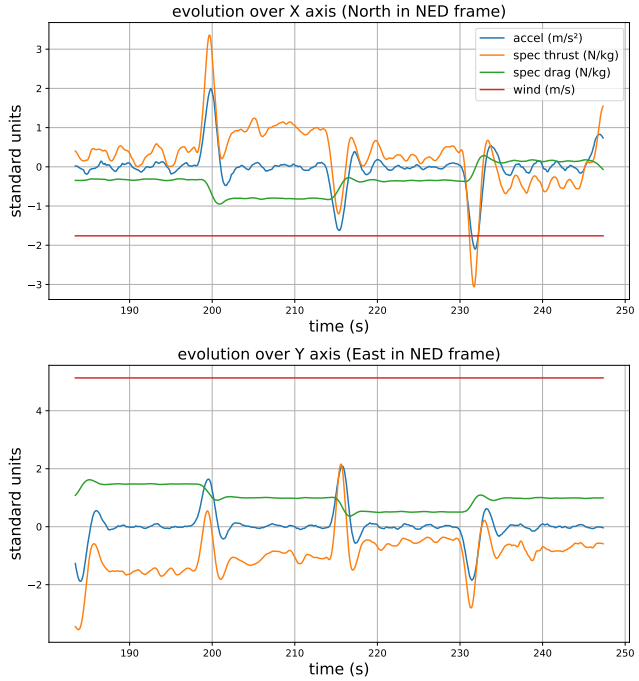


Figure 10: Evolution of acceleration, specific drag, specific thrust and estimated wind during a square trajectory in outdoor flight.

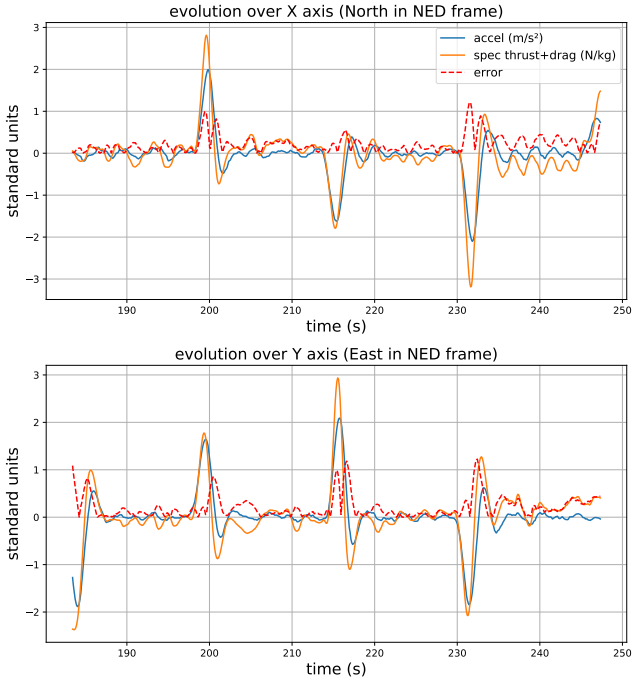


Figure 11: Evolution of equilibrium error during a square trajectory in outdoor flight.

#	mass calib (kg)	k calib	k/m calib	k/m estimated	W_x est. (m/s)	W_y est. (m/s)
1	0.547	0.153	0.279	0.181	-1.44	2.84
2	0.867	0.180	0.208	0.0999	-5.71	1.56
3	0.897	0.230	0.256	0.157	-7.02	3.09

Table 3: Experimental results for drag over mass coefficient and wind components for three different configurations: 1 - light frame, 2 - heavy frame, 3 - spherical body

The Table 3 is summarizing the outdoor experimental flights. Comparing the estimated and expected values, it is clear that this method, although giving realistic numbers, is not yet reaching the level of accuracy that one can expect for application to wind estimation.

Several hypothesis can be made to explain these bad results. First, the main assumption in the model is that the wind is constant, which was definitely not case during the experiment. Even if the flight had been performed to a calm day, a full square takes about 1 minutes. During that time, the wind is likely to have some gust. Reducing the flight time requires either a faster flight or a smaller square. The quadrotor will then face stronger or more frequent accelerations, which are the flight phases where the matching is the worst. The tests with a flight speed of 5 m/s (not presented) are producing even worse results.

Another possibility is that the pre-filtering method is not adapted and is introducing phase shift, although all data are filtered with the same parameters.

Finally, the input data, especially the attitude estimated from the onboard IMU, is very likely to have bias and delays compared to the perfect simulation data. Improving the state estimation filters should greatly improve the results.

5 Conclusion and Future Work

The aim of this study was first to investigate the influence of mass and shape on the drag of a quadrotor, and second to propose a method to estimate this drag coefficient from flight data. This parameter is crucial for wind estimation, but can also be used to improve the guidance and navigation. The conclusion for the tested model is that a light version is beneficial for the drag estimation. However, a shape generating drag is also interesting, thus a good compromise for a future experiment would be to find a structure generating drag at a low cost for the weight, for instance by using inflatable elements. Concerning the coefficient estimation, the method based on a least-square optimization, has proven to work in simulation but is facing difficulties with real flight data. There are several reasons for that, but the noise pre-filtering and the validity of assumption of constant wind during the complete procedure are most likely the critical points to address in a future work. For instance, the second point can be mitigated by flying early in the morning on a calm day, when there is almost no wind.

http://www.imavs.org/

Another approach would be to integrate the drag coefficient to the state vector of a Kalman filter estimating the wind and airspeed. Compared to the work in [13], where the filter was a linear Kalman filter, introducing the drag coefficient would have two consequences: first the dynamic becomes non-linear, which requires an Extended Kalman filter formulation, and second the new state element is not always observable, it would depend on the current speed and acceleration of the system as discussed in [11]. Our goal for a future study is to establish the observability criteria so that the state, at least for the drag coefficient, is propagated only when observable and kept as a constant parameter otherwise. The main interest of a Kalman filter is that it can cope with variations of mass and shape during the flight without the need of specific calibration procedure. This could be applied for drones used in package delivery scenarios, while other methods are more adapted to frame with fixed characteristics such as the ones used for atmospheric science.

http://www.imavs.org/

ACKNOWLEDGEMENTS

This work have been done using the preliminary work and calibration experiments of Victor Gravis, Théo Le Gall and Ziyad Baya, students at Enac, during their second year project under the supervision of the authors.

REFERENCES

[1] Stéphanie Mayer, Gautier Hattenberger, Pascal Brisset, Marius Jonassen, and Joachim Reuder. A "no-flow-sensor" wind estimation algorithm for unmanned aerial systems. *International Journal of Micro Air Vehicles*, 4(1):pp 15–30, March 2012.

[2] A. C. Kroonenberg, T. Martin, M. Buschmann, J. Bange, and P. Vörsmann. Measuring the wind vector using the autonomous mini aerial vehicle m2av. *Journal of Atmospheric and Oceanic Technology*, 25:1969–1982, 2008.

[3] 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.

[4] Jean-Philippe Condomines, Murat Bronz, Gautier Hattenberger, and Jean-François Erdelyi. Experimental Wind Field Estimation and Aircraft Identification. In *IMAV 2015: International Micro Air Vehicles Conference and Flight Competition*, Aachen, Germany, September 2015.

[5] Carl A. Wolf, Richard P. Hardis, Steven D. Woodrum, Richard S. Galan, Hunter S. Wichelt, Michael C. Metzger, Nicola Bezzo, Gregory C. Lewin, and Stephan F.J. de Wekker. Wind data collection techniques on a multirotor platform. In *2017 Systems and Information Engineering Design Symposium (SIEDS)*, pages 32–37, 2017.

[6] Fabrizio Schiano, Javier Alonso-Mora, Konrad Rudin, Paul Beardsley, Roland Siegwart, and Bruno Sicilianok. Towards estimation and correction of wind effects on a quadrotor uav. In *IMAV 2014 : International Micro Air Vehicle Conference and Competition 2014*, pages 134 – 141, Delft, 2014. International Micro Air Vehicle Conference and Competition 2014 (IMAV 2014). International Micro Air Vehicle Conference and Competition 2014 (IMAV 2014); Conference Location: Delft, Netherlands; Conference Date: August 12-15, 2014.

[7] Javier González-Rocha, Craig A. Woolsey, Cornel Sultan, and Stephan F. J. De Wekker. Sensing wind from quadrotor motion. *Journal of Guidance, Control, and Dynamics*, 42(4):836–852, 2019.

[8] L.N.C. Sikkel, G.C.H.E. de Croon, C. De Wagter, and Q.P. Chu. A novel online model-based wind estimation approach for quadrotor micro air vehicles using low cost mems imus. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2141–2146, 2016.

[9] Patrick P. Neumann and Matthias Bartholmai. Real-time wind estimation on a micro unmanned aerial vehicle using its inertial measurement unit. *Sensors and Actuators A: Physical*, 235:300–310, 2015.

[10] Matthew Marino, Alex Fisher, Reece Clothier, Simon Watkins, Samuel Prudden, and Chung Sing Leung. An evaluation of multi-rotor unmanned aircraft as flying wind sensors. *International Journal of Micro Air Vehicles*, 7(3):285–299, 2015.

[11] Robert C. Leishman, John C. Macdonald, Randal W. Beard, and Timothy W. McLain. Quadrotors and accelerometers: State estimation with an improved dynamic model. *IEEE Control Systems Magazine*, 34(1):28–41, 2014.

[12] Kilian Meier, Richard Hann, Jan Skaloud, and Arthur Garreau. Wind estimation with multirotor uavs. *Atmosphere*, 13(4), 2022.

[13] Gautier Hattenberger, Murat Bronz, and Jean-Philippe Condomines. Estimating wind using a quadrotor. *International Journal of Micro Air Vehicles*, 14:17568293211070824, 2022.

[14] 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.

Improving the computational efficiency of ROVIO

S.A. Bahnam*, C. de Wagter, and G.C.H.E de Croon
Delft University of Technology, Kluyverweg 1, Delft

ABSTRACT

ROVIO is one of the state-of-the-art mono visual inertial odometry algorithms. It uses an Iterative Extended Kalman Filter (IEKF) to align features and update the vehicle state simultaneously by including the feature locations in the state vector of the IEKF. This algorithm is single core intensive, which allows using the other cores for other algorithms, such as object detection and path optimization. However, the computational cost of the algorithm grows rapidly with the total number of features. Each feature adds three new states (a 2D bearing vector and inverse depth), leading to bigger matrix multiplications which are computationally expensive. The main computational load of ROVIO is the iterative part of the IEKF. In this work, we reduce the average computational cost of ROVIO by 40% on an NVIDIA Jetson TX2, without affecting the accuracy of the algorithm. This computational gain is mainly achieved by utilizing the sparse matrices in ROVIO.

http://www.imavs.org/

1 INTRODUCTION

Visual Inertial Odemetry (VIO) and Simultaneous Localization And Mapping (SLAM) are popular methods to navigate in GPS-denied environments. However, Micro Air Vehicles (MAVs) with extreme Size, Weight, and Power (SWaP) restrictions do not have enough computational power to do onboard loop closure computation. Especially, if the MAV has multiple computational tasks, such as object detection and path planning, it is important that computational effort for VIO is minimal. Moreover, minimal computation time and latency are important for high-speed flight, as in autonomous drone racing.

The most common VIOs are either monocular or stereo. Stereo VIO has the advantage of being able to triangulate features to immediately get a depth estimation for new features. Even though it requires an extra step (stereo matching), compared to mono VIO, it does not have to be computationally more expensive [1]. However, it requires an accurate stereo calibration and adds the weight of an extra camera. Furthermore, for drones with a smaller stereo baseline, the resolution of the depth estimation is also smaller.

Monocular VIO is preferable for MAVs with extreme SWaP constraints, as it requires only a single camera. The state-of-the-art filter-based mono VIOs are ROVIO [2] and MSCKF VIO [3]. ROVIO uses a patch-based direct method to align features and estimate the state in an Iterative Extended Kalman Filter (IEKF). Whereas, MSCKF tracks features and updates the state each time a feature is lost. A disadvantage of MSCKF is that the computational load per frame varies as it only updates if a feature is lost or when a maximum number of camera states are in the buffer. Next, there are optimization-based VIOs, like VINS-mono [4] and OKVIS [5]. However, these are generally computationally more expensive, because they optimize over a window of states.

Of the above-mentioned algorithms, ROVIO is the only algorithm that is single core intensive, which allows using the other cores for other computational tasks. Next, ROVIO is the only direct method, whereas the others are feature-based methods. The advantage of direct methods is that they are able to estimate the motion even in low-texture environments [6]. Furthermore, ROVIO is able to track features on an edge (e.g. line features) due to the initial feature location prediction it receives from the IMU-driven state propagation [2].

ROVIO has been used in various drone applications, ranging from cave exploration [7] to autonomous drone racing [8]. In a drone delivery application [9] ROVIO was considered, but SVO [10] was chosen because it is computationally cheaper and therefore has a smaller computational delay.

In this work we reduce the computational cost of ROVIO without affecting the accuracy. To be more precise we reduce the average computational time by 40% on an NVIDIA Jetson TX2. The main modifications to ROVIO are:

1. We substantially reduce the size of the Jacobian used in the IEKF of the ROVIO algorithm

2. We reduce the computational cost of the prediction step of the IEKF by exploiting the sparsity of the matrices.

The remainder of the article is organized as follows. Firstly, in Section 2 we give a short overview of ROVIO. In Section 3 it is shown what we have modified in ROVIO to reduce the computational cost. Next, the results on the EuRoC and UZH-FPV Drone Racing dataset are shown in Section 4. This is followed by the conclusion in Section 5.

2 RELATED WORK

ROVIO mainly differs from other VIOs by using an IEKF that uses photometric errors of patches as an innovation term in the filter update step. This means that the feature alignment is done simultaneously with the state update. The features

*Email address: S.A.Bahnam@tudelft.nl

http://www.imavs.org/

are included in the state vector where each feature has a 2D bearing vector and an inverse distance parameter. Furthermore, the state vector includes 21 other states: robocentric position, velocity and attitude of IMU (9), accelerometer and gyroscope biases (6), and linear and rotational part of the the IMU-camera extrinsics (6). Therefore, the state vector has a size of $n = 21 + 3 \cdot m$, where m is the maximum number of features (25). The computation for each new image frame can be described in 3 steps. Firstly, the state is predicted using the IMU data between time t and $t-1$. Next, for each feature, the state vector is iteratively updated until the feature is matched (or discarded as an outlier). Lastly, new features are added when the number of tracked features drops below a certain threshold (of $0.8 \cdot m$).

Adding new features has a low average computational cost, but can result in computational peaks. The reason for this is that only new features are added when the number of features is dropped below 80% maximum number of features. Therefore, in most frames no features need to be detected, which result in a low average computation time. However for frames which adds new features, it requires additional computation for feature detection next to the IEKF computation, which result in a computational peak in ROVIO. When looking only to the new feature selection part of ROVIO, the computation of the Shi-Tomasi score [11] of all candidates is the computational most expensive part.

In most VIO evaluations on benchmarks, the VIO performance is not affected by computational peaks, because a camera buffer is used in the VIO. However, the control performance of applications, such as autonomous drone racing, is affected by delay. Therefore, one would like to decrease the computational delay. For this reason, one could decide to reduce the camera buffer in such applications. However, this comes at the cost that the accuracy of the VIO may decrease or in worst case that the filter diverges if the peak processing time is too high. In [8] it is reported that ROVIO was processed at 35 Hz, but the total delay was 130 ms. It is also mentioned that the main contribution to the total delay was interfacing with the camera and running the VIO. In this work we reduce the computational cost of ROVIO to increase the execution frequency and reduce delays..

3 METHOD

3.1 Prediction step of IEKF

In the prediction step of the IEKF, the states are propagated and the covariance is estimated using the IMU data. ROVIO uses the average IMU data between two frames in order to compute the prediction step only once every frame and reduce the computational cost without a notable performance loss. The covariance matrix computation is computationally most expensive as it involves $n \times n$ matrix multiplications and can be calculated with Equation 1.

$$P_k^- = F_{k-1} \cdot P_{k-1}^+ \cdot F_{k-1}^T + G_{k-1} \cdot W_{k-1} \cdot G_{k-1}^T \quad (1)$$

, in which P is the covariance matrix, $k-1$ and k are before and after the state prediction, respectively. F is the system transition matrix, G is the noise input matrix and W is the continuous time noise covariance. Each matrix here has a size of $n \times n$, where n is equal to $21 + 3 \cdot m$, where m is the maximum number of features. Therefore, the multiplication of the matrices is computationally expensive. However, matrix F, G, and W are all sparse matrices. W is a constant diagonal matrix, where all entries are the estimated noise variances from the input. F and G can be found in Equation 2 and 3, respectively. For more details how F and G are constructed, see [2].

$$F = \begin{bmatrix} I_{3\times3} & & & \\ & B_{15\times12} & 0_{15\times6} & 0_{21\times f} \\ 0_{n-3\times3} & 0_{6\times12} & I_{6\times6} & \\ & B_{f\times12} & 0_{f\times6} & BD_{f\times f} \end{bmatrix} \quad (2)$$

$$G = \begin{bmatrix} & 0_{3\times3} & & \\ D_{12\times12} & B_{3\times3} & 0_{15\times6} & 0_{21\times f} \\ & 0_{6\times3} & & \\ & B_{3\times3} & & \\ 0_{n-12\times12} & 0_{6\times3} & D_{6\times6} & \\ & B_{f\times3} & 0_{f\times6} & BD_{f\times f} \end{bmatrix} \quad (3)$$

, in which B is a block matrix, BD a block diagonal (3×3 block diagonal for F and G), D is a diagonal matrix and f is the number of all feature states ($3m$).

3.2 Iterative update of IEKF

The difference between an EKF and IEKF is that the update step is performed multiple (j) iterations until the update is small/converged, the measurement is discarded (detected as outlier), or the maximum number of iterations (20 for the original settings) is reached. ROVIO iteratively updates the state vector for each feature candidate (i) separately.

Each iteration requires big matrix multiplications to calculate the Jacobian, Kalman gain, update vector, and the covariance matrix of the state update. The big matrices are sparse, however the used MatrixXd from the Eigen library [12] in ROVIO does not perform sparse matrix multiplication efficiently.

A feature candidate is generated using the the predicted feature location and its covariance. This is done at least once and a maximum of 3 times (for the original settings) per detected/tracked feature on the previous frame. Each time the state vector is updated as in Equation 4.

$$x = x + P_{i_{n\times n}}^- \cdot -c_{i_{2\times n}}^T \cdot Py_i \cdot dy_i \quad (4)$$

, in which $P_{i_{n\times n}}$ is the state covariance matrix, c_i is the pixel location of the feature inserted in a $2 \times n$ zero matrix, which is done to get the correct size for matrix multiplication. dy_i is a 2×1 vector, which depends on the eigenvalues of the Py_i. And Py_i is a 2×2 matrix and can be calculated with Equation 5.

$$Py_i = -c_{i_{2\times n}} \cdot P^-_{i_{n\times n}} \cdot -c^T_{i_{2\times n}} \tag{5}$$

Since c_i only contains information about the pixel location we can save computation time. Depending on which feature is processed, we use a certain 2×2 block of those matrices. Therefore, we can modify Equation 4 and 5 to Equation 6 and 7, respectively.

$$x = x + P^-_{i_{n\times 2}} \cdot -c^T_{i_{2\times 2}} \cdot Py_i \cdot dy_i \tag{6}$$

$$Py_i = -c_{i_{2\times 2}} \cdot P^-_{i_{2\times 2}} \cdot -c^T_{i_{2\times 2}} \tag{7}$$

Next, a multilevel patch is extracted (a patch of 6×6 on two image levels) for each feature candidate. The Jacobian of the feature is calculated based on the multilevel patch gradient, adaptive light condition parameters and the (distorted) feature pixel location. The gradient of the previous image is used, therefore the multilevel patch gradient is constant for all iterations per feature in a single frame. However, the original ROVIO algorithm unnecessarily recomputes the gradient at every iteration of each feature per frame (i, j, k). We store the gradient of the patch of a feature on the first iteration of each frame. Therefore, we only need to calculate the gradient once per feature per frame (i, k).

In ROVIO the 2x2 Jacobian is inserted in a $2 \times n$ zero matrix, with n being the size of the state vector. This is done to allow matrix multiplications with the $n \times n$ covariance matrix. However, this is very inefficient as many zero multiplications are involved. Therefore, we extract the useful information in the 2×2 block Jacobian for calculations. This allows us to use smaller blocks for the covariance matrix as well. The original code of ROVIO calculates the 2×2 matrix $Py_{i,j}$ using Equation 8. We propose to calculate $Py_{i,j}$ as in Equation 9.

$$Py_{i,j} = H_{i,j_{2\times n}} \cdot P^-_{i_{n\times n}} \cdot H^T_{i,j_{2\times n}} + R \tag{8}$$

$$Py_{i,j} = H_{i,j_{2\times 2}} \cdot P^-_{i_{2\times 2}} \cdot H^T_{i,j_{2\times 2}} + R \tag{9}$$

, in which $P^-_{i_{2\times 2}} = P^-_i.block(21 + 3i, 21 + 3i, 2, 2)$ is a 2×2 block of the covariance matrix of the prediction step, where i is the index of the processed feature in the state vector. $H_{i,j_{2\times 2}}$ is the Jacobian without zeros. R is the measurement noise matrix with size 2×2. Note, that the computational time of our proposed method is independent of the size of the state vector and the original method is $O(n^3 + n^2)$. Next, the Kalman gain (of size $n \times 2$) is calculated with the original code as in Equation 10. We propose to modify it to Equation 11.

.

$$K_{i,j} = P^-_{i_{n\times n}} \cdot H^T_{i,j_{2\times n}} \cdot \left(Py_{i,j_{2\times 2}}\right)^{-1} \tag{10}$$

$$K_{i,j} = P^-_{i_{n\times 2}} \cdot \left(H^T_{i,j_{2\times 2}} \cdot \left(Py_{i,j_{2\times 2}}\right)^{-1}\right) \tag{11}$$

Since K_i, j is of size $n \times 2$ we have to use all rows of our covariance matrix, but we only need two columns. Furthermore, we first do the 2×2 matrix multiplication as this will save computational work. This reduce the computational cost from $O(n^3 + n)$ to $O(n)$.

The original code uses Equation 12 to compute the update vector and we propose Equation 13 instead.

$$\Delta x_{i,j} = (x^-_i \boxminus x^+_{i,j}) - K_{i,j} \cdot \left(z_{i,j} + H_{i,j_{2\times n}} \cdot (x^-_{i_{n\times 1}} \boxminus x^+_{i,j_{n\times 1}})\right) \tag{12}$$

$$\Delta x_{i,j} = (x^-_i \boxminus x^+_{i,j}) - K_{i,j} \cdot \left(z_{i,j} + H_{i,j_{2\times 2}} \cdot (x^-_{i_{2\times 1}} \boxminus x^+_{i,j_{2\times 1}})\right) \tag{13}$$

, in which the $\boxminus$ is the boxminus operator. The computational cost is reduced from $O(n^2 + n)$ to $O(n)$.

Furthermore, the Jacobian, H_j, and the measurement, z_j, (difference of the multilevel patches) are computed twice per iteration in the original ROVIO code. This is modified, such that it is only calculated once per iteration.

4 RESULTS

We test the modified and original ROVIO on the EuRoC [13] dataset and on the UZH-FPV Drone Racing Dataset [14]. We run the algorithms on an NVIDIA Jetson TX2, which has a dual-core Denver 2 64-Bit CPU and a quad-core ARM Cortex-A57. We use the tool of [15] to get the RMS of the APE. All trajectories are aligned with the ground truth, optimizing position and yaw only, which is proposed for mono VIOs in [15].

Firstly, we show the difference in computation time, due to the modifications. Next, we show that the modified ROVIO does get a similar trajectory estimation when processing all frames on the EuRoC dataset. Finally, we reduce the camera buffer and compare the accuracy of the original and modified algorithm on the UZH-FPV Drone Racing Dataset.

4.1 EuRoC

We compare the original ROVIO and modified ROVIO on the EuRoC dataset. We exclude sequence Machine Hall 2, because ROVIO (with original parameters) diverges on this sequence. Furthermore, we exclude the first 21 seconds of the sequence Machine Hall 1, in order to ensure ROVIO initializes correctly (and similarly) for the original and modified

http://www.imavs.org/

Table 1: RMS of the APE of the original and modified ROVIO on the EuRoC dataset after aligning position and yaw with the ground truth. In bold we show when the modified ROVIO matches the original ROVIO either on the laptop or TX2.

		MH01[1]	MH03	MH04	MH05	V101	V102	V103	V201	V202	V203
Laptop	Original	0.346	0.449	0.819	1.307	0.161	0.133	0.225	0.239	0.346	0.249
	Modified	**0.346**	**0.449**	**0.819**	**1.307**	**0.161**	0.196	**0.176**	**0.242**	**0.394**	**0.251**
TX2	Original	0.346	0.398	0.819	1.307	0.156	0.175	0.176	0.242	0.394	0.251
	Modified	**0.346**	**0.449**	**0.819**	1.311	**0.161**	**0.175**	**0.225**	**0.242**	**0.394**	0.237

version. Furthermore, we check for both algorithms that they initialize at the exact same (IMU) timestamp and process the same number of frames.

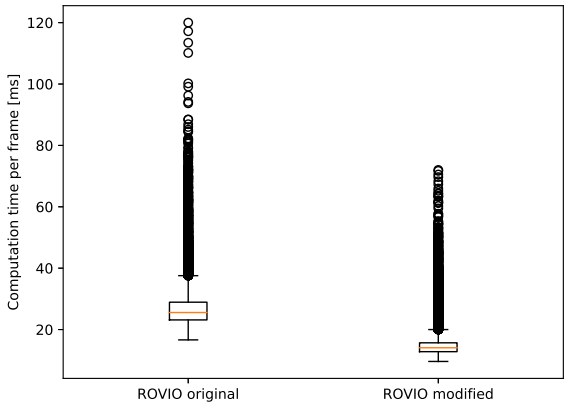


Figure 1: Computation time per frame on the EuRoC dataset (all sequences except MH02) on an NVIDIA Jetson TX2

In Figure 1 the computation time on an NVIDIA Jetson TX2 can be found for the original and modified ROVIO. It can be seen that the average and the maximum computational time per frame is reduced. The original algorithm has an average computation time per frame of $25.6\ ms$ and the modified algorithm this is reduced to $14.1\ ms$. It can be seen that the modified algorithm still have computational peaks. Those mainly correspond to frames where new features are added. The main computational cost of the feature detection is the computation of the Shi-Tomasi score of the (usual) many detected candidates.

In Table 1 it can be seen that the accuracy sometimes differs when we run the original and modified algorithm. Also it differs when we run the same algorithm on different hardware. The reason for this are rounding errors, which sometimes can lead to a feature at the image border being seen as "out of frame". This may result in different features being detected and tracked, which results in a change in accuracy. However, it can be seen that the modified algorithm always has the same accuracy as the original algorithm either on the laptop or Nvidia Jetson TX2. Except for V102 on the Laptop and V203 on the TX2.

[1]The first 21 seconds are excluded of this sequence

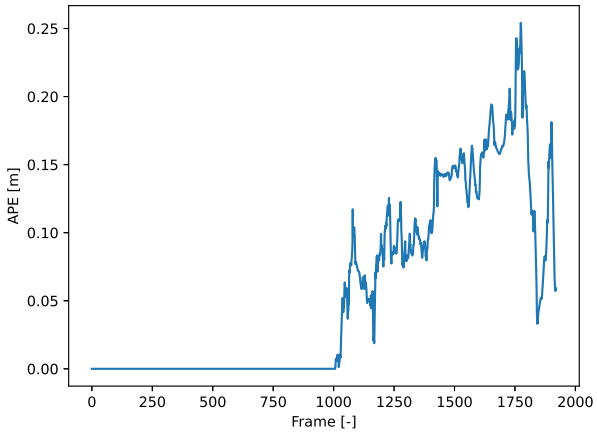


Figure 2: APE of the modified ROVIO w.r.t. the original ROVIO on sequence V203 of the EuRoC dataset on an NVIDIA Jetson TX2

The difference on V102 on the laptop is quite big. However, when we change the feature detection threshold from 0.8 to 0.999 we get and average RMSE of $0.135\ m$ for sequence V102 for modified and original algorithms on both hardware. By changing this threshold new features are added every time a feature is inactive (not tracked), instead of when the number of tracked features drops below 80% of the maximum number of features. Therefore, the change in feature set is reduced.

The modified ROVIO on the TX2 gets a slightly lower RMSE on sequence V203, but the difference is small. In Figure 2 the difference between the original and modified algorithm on the TX2 are shown. The reason for the difference is that at the 1006th frame one feature is lost in the modified algorithm, however that same feature is tracked with the original algorithm. The reason for this is a rounding error when computing Equation 7, if we use the original Equation 5, the modified algorithm matches the original ROVIO on sequence V203. However, it results in that other sequences will experience a similar rounding error (in another part of the modified algorithm).

http://www.imavs.org/

Table 2: Total frames processed per sequence using a camera buffer of 1 on the UZH-FPV Drone Racing Dataset with an NVIDIA Jetson TX2.

	03	05	06	07	09	10
Original	2124/2552	4073/4162	1589/1970	2850/3158	2051/2068	2082/2127
Modified	**2437/2552**	**4157/4162**	**1935/1970**	**3142/3177**	**2066/2068**	**2124/2127**

Table 3: RMS of the APE of the original and modified ROVIO on the UZH-FPV dataset after aligning position and yaw with the ground truth.

	03	05	06	07	09	10
Original	1.47	0.72	0.50	1.08	**0.38**	0.62
Modified	**0.97**	**0.59**	**0.48**	1.08	0.52	**0.58**

In order to verify that the modified matrix multiplication is done correctly, we use the Frobenius norm of the matrices for a fuzzy comparison (Eigen::isApprox) as in Equation 14.

$$||V - W||_F = p \cdot min(||V||_F, ||W||_F) \tag{14}$$

, in which V is the matrix calculated using the proposed equations, W is the result using the original matrix multiplication from ROVIO. We set p to 10^{-12}, because this is the default value for a fuzzy comparison with double precision matrices. We do this test for all proposed modifications separately. All modified equations pass the test, except the update vector calculation from Equation 13. In all sequences it returns 0.1% false or less. This is because the magnitude of the update vector is sometimes small (when the prediction is close to the measurement), which results in a very strict comparison.

4.2 UZH-FPV Drone Racing Datset

We run the Original and modified version of ROVIO as well on the UZH-FPV dataset [14]. We are using sequence 03, 05, 06, 07, 09 and 10 of the forward facing camera indoor, because those sequences contain the ground truth of the trajectory. We reduce the camera state buffer to 1, which means that a frame (the oldest) is dropped when the algorithm is not fast enough to process two consecutive frames. We do the evaluation only on the NVIDIA Jetson TX2, because the laptop is fast enough to process all frames. This type of evaluation would be realistic for cases where the state estimation delay is of importance. Which could be for example autonomous drone racing, because a delay in the state estimation affects the control performance. We do not change any other parameters of the algorithm.

In Figure 3 the computation time on the UZH-FPV dataset can be seen. The average computation time for the original algorithm is $31.5\ ms$ and for the modified $17.4\ ms$. Even though, the image resolution is smaller on the UZH-FPV dataset (640×480) than on EuRoC (752×480), the computation time is higher on the UZH-FPV dataset. We think this is because the motion on the UZH-FPV is bigger, which

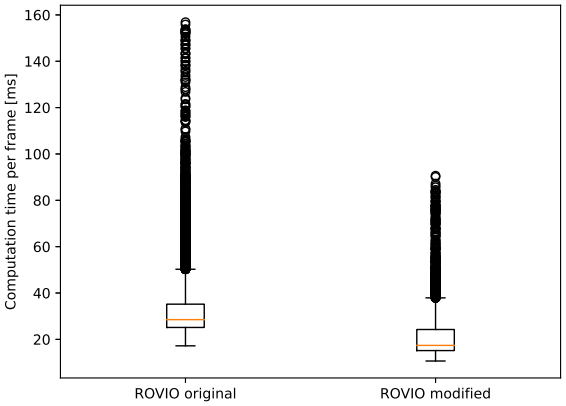


Figure 3: Computation time per frame on (all sequences of) the FPV dataset on an NVIDIA Jetson TX2

requires to use more iterations in the IEKF to align the features.

In Table 2 it shows how many frames out of the total frames are processed for the modified and original ROVIO. Note, that the full (rosbag) sequence is used here. It can be seen that fewer frames are thrown away for the modified ROVIO than the original ROVIO. The reason for this is the computational gain from the sparse matrix operations. It can also be seen that almost all frames are processed for the modified algorithm. The missed frames are expected to come from frames where new features are added, because this adds additional computation in a frame.

The RMSE of the original and modified ROVIO can be found in Table 3. Again we align the the estimated trajectory with the ground truth by optimizing position and yaw only. Note that for the UZH-FPV dataset that the ground truth is only available for part of the sequences. The modified algorithm is more accurate for most sequences, which is expected since it was able to process more frames. The difference in accuracy is most visible in sequence 03. In this sequence there is a big difference in the number of processed frames. However, processing more frames does not guarantee a lower RMSE. On sequence 07 the RMSE of the modified ROVIO is the same as the original algorithm, but the modified algorithm processed almost 300 more frames. On sequence 09, the original algorithm has even a smaller RMSE, however sequence 09 the original ROVIO did not drop many frames either.

http://www.imavs.org/

5 CONCLUSION

We have made ROVIO computationally more efficient, mainly by utilizing the sparse matrices. The accuracy is the same as the original ROVIO, but requires 40% less computation time on an NVIDIA Jetson TX2. The computational gain depends on the number of features used, because the computational cost of the modified ROVIO is less dependent on the size of the state vector compared to the original ROVIO.

Furthermore, it was shown that the modified ROVIO has a higher accuracy when using a camera buffer of one. However, the difference was small for most sequences. We expect that the difference is bigger when running on a computationally more limited device or when increasing the maximum number of features.

The computational peaks are reduced as well, but they are still there. To mitigate the computational peaks, we recommend do modify the feature selection method. To be more specific: reducing the number of feature candidates of which the Shi-Tomasi score is computed. This could for example be done by using a feature mask around tracked features or using an adaptive FAST threshold. However, this will affect the accuracy of ROVIO and therefore we have not included this in this work.

REFERENCES

[1] Ke Sun, Kartik Mohta, Bernd Pfrommer, Michael Watterson, Sikang Liu, Yash Mulgaonkar, Camillo Taylor, and Vijay Kumar. Robust stereo visual inertial odometry for fast autonomous flight. *IEEE Robotics and Automation Letters*, PP, 11 2017.

[2] Michael Bloesch, Michael Burri, Sammy Omari, Marco Hutter, and Roland Siegwart. Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback. *The International Journal of Robotics Research*, 36:1053–1072, 09 2017.

[3] Anastasios I. Mourikis and Stergios I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *2007 IEEE International Conference on Robotics and Automation, ICRA'07*, Proceedings - IEEE International Conference on Robotics and Automation, pages 3565–3572, November 2007.

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

[5] 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, 02 2014.

[6] Guoquan Huang. Visual-inertial navigation: A concise review. *2019 International Conference on Robotics and Automation (ICRA)*, pages 9572–9582, 2019.

[7] Mihir Dharmadhikari, Huan Nguyen, Frank Mascarich, Nikhil Khedekar, and Kostas Alexis. Autonomous cave exploration using aerial robots. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 942–949, 2021.

[8] Philipp Foehn, Dario Brescianini, Elia Kaufmann, Titus Cieslewski, Mathias Gehrig, Manasi Muglikar, and Davide Scaramuzza. Alphapilot: autonomous drone racing. *Autonomous Robots*, 46, 01 2022.

[9] Gino Brunner, Bence Szebedy, Simon Tanner, and Roger Wattenhofer. The urban last mile problem: Autonomous drone delivery to your balcony. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1005–1012, 2019.

[10] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, and Davide Scaramuzza. Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2):249–265, 2017.

[11] Jianbo Shi and Tomasi. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[12] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[13] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35, 01 2016.

[14] 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 *2019 International Conference on Robotics and Automation (ICRA)*, pages 6713–6719, 2019.

[15] Zichao Zhang and Davide Scaramuzza. A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2018.

APPENDIX A: CODE

A branch of the updated code can be found at: http://www.github.com/sbahnam/rovio2

http://www.imavs.org/

Fast NIR-Single-Pixel-Imaging enhancement under scattering environment

Carlos A. Osorio Quero*, Daniel Durini, Jose Rangel-Magdaleno, Jose Martinez-Carranza and Ruben Ramos-Garcia
Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), 72840, Mexico.

http://www.imavs.org/

ABSTRACT

The captured images in scattering scenarios are a challenge to solve for the vision systems for Unmanned Flight Vehicles (UFVs). Currently, research has been conducted on the use of sensors in the near-infrared (NIR) part of the spectra as an option for the RGB camera in scenarios with higher scattering (fog or rain), because to the diameter of water particles is proportional to the wavelength, then Mie scattering occurs with that the transmittance is higher. Therefore, in this work, we present a solution vision system based on single-pixel imaging at the wavelength of 1550 nm (NIR-SPI) to capture 2D images in a scattering environment (Fog and Rain). For which we evaluate the performance of the 2D image reconstruction in controlled laboratory settings using different compressive sensing (CS), Fourier single-pixel imaging (FSI), and Hadamard-single-pixel-imaging (HSI) reconstruction methods fused with dark channel prior (DCP) dehazing methods to improve 2D image quality. This work will determine which strategy of processing and imaging is more suitable in terms of quality image (EPI), entropy and contrast, and processing time to suit the Unmanned Flight Vehicles (UFVs).

1 INTRODUCTION

In the last year, the use the Unmanned Flight Vehicles (UFVs) increase, covering different applications in outdoor environments as infrastructure inspection [1], rescue operations [2], and surveillance [3]. For which is needs to be captured image under different conditions, light and weather. However, bad weather images captured show some degradation effects such as blurriness and low contrast. Due to light traveling through a medium that contains suspended water-particles, it is exponentially attenuated through two processes: absorption and scattering [4]. Being difficult to distinguish objects in the environment. In such scenarios, the performance of the vision system based on RGB (Red-Green-Blue) is limited, in the visible spectrum we meet in the Rayleigh regime [5] with a D diameter water-particles is $D << \lambda$, where λ is the wavelength , in this case the attenuation the light is highest decreases the contrast level , while for wavelength Near-infrared (NIR) $D \sim \lambda$ is in the Mie regime [6], where the effects of the light attenuation and scattering are fewer in comprised with the visible spectrum, being the use of NIR sensor a solution for increase performance vision systems of the UFVs [7].

Therefore, the solution proposed in this work is a system vision on NIR based on a Single-Pixel Camera (SPC) [8], an architecture that uses a single detector as an alternative to the traditional cameras that capture the scene directly using a sensor array (see Fig. 1), and has these two essential advantages: (i) it provides the means to acquire the image in the compressed form, reducing the processing time; and (ii) it is suitable to perform imaging to different wavelengths, with that we can take advantage from a wider part of the electromagnetic spectrum, which is ideal for application with UFV. This system allows capturing the image in the NIR spectrum, where the attenuation environment is the lowest visible, and the image obtained can be enhanced through of used Dark Channel Prior (DCP) method [9]. From background light (BL) estimation and transmission map (TM) over image, single-pixel imaging (SPI) is possible to refine the image by removing the hazy effects in the image.

To summarize, the main contributions and limitations of this paper are as follows:

- We developed a SPC NIR-SPI with active illumination at a wavelength of 1550 nm with a resolution size of 8x8 and implement three type reconstruction algorithms based on compressive sensing (CS), Fourier single-pixel imaging (FSI), and Hadamard-single-pixel-imaging (HSI).

- Implementation the Dark Channel Prior (DCP) method over images SPI and evaluation of laboratory the NIR-SPI system in the test chamber.

- Evaluation of the CS, FSI, and HSI methods 2D reconstruction of SPI in a simulated environment applying the proposed DCP algorithm, in terms of Edge Preservation Index EPI [10], and processing time. To determine which method best adapts the conditions, of the scattering environment.

*Email address(es): caoq@inaoep.mx

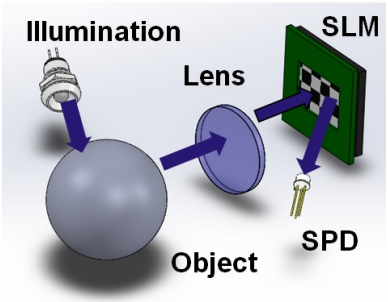


Figure 1: Schematic generation single-pixel imaging:the object illuminated by a light source and the light reflected by it gets directed through a lens onto an SLM, where is capture by the SPD [7].

2 SINGLE-PIXEL IMAGING (SPI) 2D RECONSTRUCTION

Single-pixel imaging is based on the projection of spatially structured light patterns by using light modulation devices such as: (i) Spatial light modulators (SLM) that allow light to be modulated in terms of amplitude, phase, or wave polarization in space and time, (iii) Digital micromirror devices (DMD) that operate as digital on/off modulators of the reflected light on the micromirror array, through the change of the tilt angle it is possible to reflect or obstruct light projecting a light pattern modulate over the scene, (iii) LED matrix as light digital on/off modulation, that allow faster changes the light patterns than SLM and DMD devices. The reflected light is focused on a photodetector with no spatial information S_i (see Fig. 1). From the correlations between the patterns $\Phi_i(x,y)$ and S_i, we can reconstructed the image $O,(x,y)$, where the reconstructed image is obtained as the product of the measured single S_i and the corresponding structured pattern that originated defined by Eq. (1) [11].

$$O\,(x,y) = \alpha \sum_{x=1}^{M}\sum_{y=1}^{N} S_i \Phi_i\,(x,y) \tag{1}$$

For reconstruction SPI, we can apply different as compressive sensing (CS), Fourier single-pixel imaging (FSI), and Hadamard-single-pixel-imaging (HSI) methods, which use transformations or iterative algorithms [8]. In FSI and HSI, both acquire the spectrum, the object image and reconstruct the object image by applying an inverse transform. For FSI, the sinusoidal patterns (see Fig. 2b), defined by Eq. (2) [12].

$$\Phi_{iFSI}(x,y) = a + b \cdot cos(2\pi x + 2\pi y + \phi) \tag{2}$$

$$R(u,v)_{FSI} = [R_0(u,v) - R_\pi(u,v)] + j[R_{\pi/2}(u,v) - R_{3\pi/2}(u,v)] \tag{3}$$

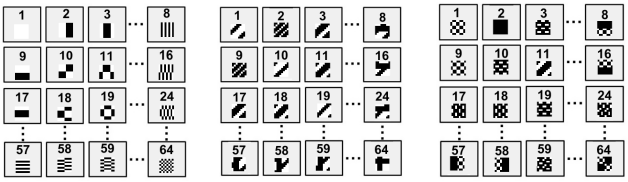


Figure 2: Patterns SPI: a) Hadamard patterns basic [7], b) FSI patterns [12], c) HSI patterns [8].

Where the variable a is the average pattern intensity and the variable b is the contrast [12]. The projection of the FSI patterns varies with the phases ($\phi = 0, \phi = \pi/2, \phi = \pi, \phi = 3\pi/2$). We can obtain a value of intensity light over a detector with a spectrum of the object $R(u,v)_{FSI}$ expressed by Eq. (3). The object image $O_{FSI}(x,y)$ can then be reconstructed by applying the IFT2 $O_{FSI}(x,y) = F^{-1}(R(u,v)_{FSI})$.

For HSI, we used the Hadamard patterns Eq. (4) (see Fig. 2c), from the patterns projected over the object can obtain differential measurement $R(u,v)$ and the reconstruction object $O_{HSI}(x,y)$ can be obtained using an inverse Hadamard transform $O_{HSI}(x,y) = H^{-1}(R(u,v))$ [12]. To implementation the FSI and HSI patterns in DMD or LED array is must apply dither algorithm to generate binary patterns, that they can be adapt theses technologies [13].

$$\Phi_{iHSI}(x,y) = \tfrac{1}{2}(1 + H^{-1}\delta^{th}(u,v)) \qquad \delta^{th}(u,v) = \begin{cases} 1, u = u_0, v = v_0 \\ 0, otherwise \end{cases} \tag{4}$$

In the case of the CS method, the generate the Hadamard pattern Φ_i based on Sylvester's recursive matrix generation principle [8] (see Fig. 2a). For reconstruction, apply CS, we use an iterative method based on (Orthogonal Matching Pursuit) OMP-GPU [14], where it needs to Hadamard pattern Φ_i and measurements.

3 BACKGROUND

When light interacts with water-particles, various physical phenomena occur such as: reflection, refraction, absorption, and scattering. The scattering is related to the particle size (sp), and its cross-section Q_d [7], where $sp = \pi D/\lambda$, D is the particle diameter, and λ the incident photon wavelength. For small scattering particles with $D << \lambda$, the distribution of scattered light is symmetric in the forward and backward directions (Rayleigh regime). If the particle size increases, the scattered energy increases toward the incident beam, reducing the amount of scattered light in the opposite direction (Mie regime).

http://www.imavs.org/

http://www.imavs.org/

3.1 Atmospheric scattering model

The image received by a sensor from scene points is often absorbed and scattered by the medium. Therefore, an image can be defined using McCartney's atmospheric scattering model using Eq. (5) [15].

$$I(x,y) = J(x,y)TM(x,y) + BL(x,y)(1 - TM(x,y)) + n(x,y) \quad (5)$$

Where I (x, y) denotes the observed degraded image. J (x, y) denotes the scene radiance, which represents the original appearance of image BL (x, y), the global atmospheric light, and TM (x, y) is the atmospheric transmission map defined by Beer-Lambert-Bouguet law as $TM(x,y) = e^{-\beta \hat{L}}$, $-\beta$ is the extinction coefficient [9]. The term, J(x,y)TM(x,y) is called the direct attenuation indication, of the decay of the scene radiance in the medium. BL (x,y)(1-TM(x,y)) is the airlight increases as the scene depth increases, and n (x, y) denotes zero-mean Gaussian noise from the environment and sensor.

3.2 Dehazing Image

In the DCP-based dehazing algorithm [9]. For applying DCP algorithm there are six steps: (i) airlight estimation (BL), (ii) dark channel estimation $J(x,y)$, (iii) coarse transmission estimation (TM), (iv) transmission refinement, (v) scene recovery, and (vi) image equalization (see Fig. 3).

- **Airlight estimation (BL):** Obtained from defined over image $(I(x,y)^{dark})$ a local patch or area, where local entropy is measured for estimation BL Eq. (6) [9]. The local entropy value is low for regions with smooth variations, which correspond to haze-opaque regions (zone with scattering)

$$BL(x,y) = I(argmax_{x,y}(I(x,y)^{dark})) \quad (6)$$

- **Dark channel estimation:** The dark channel is first constructed from the input image J, the estimation of the dark channel is shown in Eq. (7), where J^c represents the color channel of J and $\Omega\ (x,y)$ is local path anchored, the value $J^{dark}(x,y)$ is low and tends to be zero except for the sky region [9]. The transmission in a local path $\Omega(x,y)$ is assumed to be a constant.

$$J^{dark}(x,y) = \min_{y \in \Omega(x,y)} \left(\min_{c \in \Omega(r,g,b)} J^c \right) \quad (7)$$

- **Coarse transmission estimation (TM):** The Eq. (8) [9], with a control parameter ω= 0.95, The luminance-based transmission map $TM(x,y) = e^{-\beta \hat{L}}$, $\hat{L}$ denoting the modified luminance value define as $\hat{L} = \frac{\tau}{L^*} L(x,y)$, where L(x,y) is the luminance of the input image I(x,y), τ is the depth range , and L^* denote a percentage of the luminance L(x,y) .

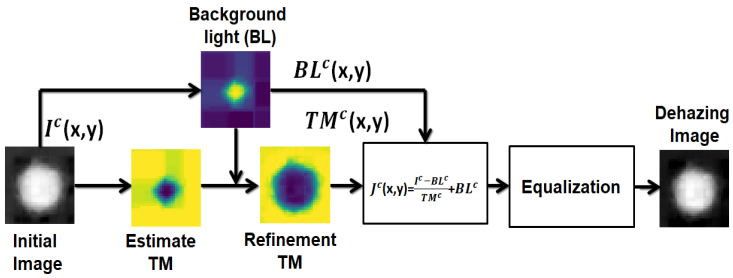


Figure 3: Overall block diagram Dark Channel Prior (DCP) method applying on single-pixel foggy environment [9].

$$TM(x,y) = 1 - \omega \min_{y \in \Omega(x,y)} \left(min \frac{I^c}{BL^c} \right) \quad (8)$$

- **Transmission refinement:** Incorrect estimation of the transmission map can lead to problems such as false texture and blocking artifacts. To improve the accuracy of the transmission map, there are methods such as Gaussian, Bilateral, Guided or Cross-bilateral filters and soft matting using minimizing energy function [9].

- **Scene recovery:** The atmospheric light and the transmission map are then obtained from the dark channel. The transmission map is further refined, and the haze-free image is finally reconstructed as Eq. (9) [9].

$$J(x,y) = \frac{I(x,y) - BL(x,y)}{max(TM(x,y), t_{th})} + BL(x,y) - \frac{n(x,y)}{TM(x,y)} \quad (9)$$

- **Image equalization:** The image recovery is equalized improvement the level contrast the new image.

For the implementation of the DCP algorithm on GPU over images SPI, we proposed the reducing computational complexity of implementation fast guided filter by transmission refinement, using the strategy of comparison score value with threshold θ_{th} score=mean(I(x,y))-var(I(x,y))$> \theta_{th}$ for air-light estimation.

4 NIR-SPI SYSTEM

The NIR-SPI vision system using structured illumination, which improves image quality in the presence of strong backlight and stray light (see Fig. 1), provided by an array of 8 x 8 NIR-LEDs emitting radiation with a peak wavelength of 1550 nm. The SPD used is an InGaAs photodiode. This active illumination approach offers several advantages, as it

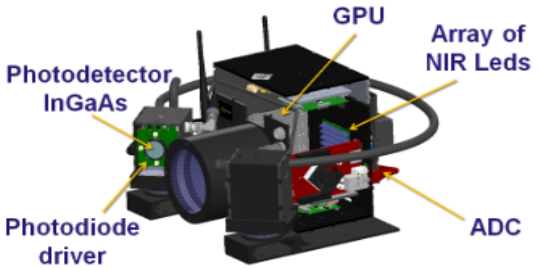


Figure 4: Proposed Single-Pixel Camera (SPC) dimension 11 x 12 x 13 cm, focal length 20 cm, weight 1.3 kg, power consumption 25 W, module photodiode, active illumination source, photodetector diode InGaAs FGA015 GPU unit and ADC [7].

can operate in different outdoor weather under low-level illumination (scenarios with dust, fog, rain, or smoke) and it is less sensitive to background radiation noise [7]. The NIR-SPI architecture proposed in this work is divided into two main parts: (i) the basic elements used to generate images through the single-pixel principle: an InGaAs photodetector (diode FGA015 @ 1550 nm), an array of emitting NIR-LEDs, and an ADC (see Fig. 4), and (ii) the subsystem in charge of processing the electrical output signal provided by the SPD module, digitized using the ADC, and the respective data processing using a Graphics Processing Unit (GPU) (see Fig. 4). The GPU unit (Jetson-Nano) is responsible for generating the Hadamard patterns and processing the converted data by the ADC, used by the Batch-OMP Algorithm [14] running in the GPU unit to generate the 2D images.

5 Experimental Results

To evaluate the Single-Pixel Camera (SPC) capabilities with active illumination using the different projection methods, we developed a testing bench that has a controlled system illumination to simulate background outdoor light and a system that can simulate the conditions of scattering environment. As test objects, we used rocks and pine seeds (see Fig. 6- Fig.7) to emulate a forest environment an explore capability capture of NIR image. In the tests, we defined a scenario foggy (water-particles diameter 3μm [7]), and levels of illumination 15 Klux (half-cloudy)-30 Klux (midday) [7], where the object test is at distance 60 cm. For reconstructing the 2D image applying the CS, FSI and HSI methods under the two light conditions. The 2D SPI image (see Fig. 6) is improved applying the DCP method (see Fig. 7- 8), as evaluation parameters of the final reconstructed images, we calculated the corresponding EPI [10], setting the goal parameters as EPI $>$ 0.5 (see Tab. 1), and time processing (see Tab. 2).

5.1 Discussion: Testing 2D NIR-SPI enhancement image

In the tests carried out to a 2D image reconstruction in a scattering environment applying the FSI, HSI, and CS methods at a measured distance 60 cm with illumination conditions of midday (see Fig. 6), and half-cloudy (see Fig. 8a). In the first test, with illumination conditions midday, we applied the DCP algorithm and obtained values of EPI $>$ 0.5 (see Table. 1) for the cases of HSI and CS with processing times between 41 and 47 ms (see Table. 2). For FSI observed a processing time fewer, with a value of EPI $<$ 0.5 showing low-capacities reconstruction and high sensitivity in scattering scenarios. In the second test, we changed the scenario by reducing the level of illumination (half-cloudy) for this scenario the contrast of the image reconstructed is reduced (see Fig. 8) , then applying the DCP algorithm, shows its high capacities for improving image quality obtained values of EPI $>$ 0.7 (see Table. 1) being the CS method the best with a processing time of 39 ms (see Table. 2), although in the case FSI presents a processing time, the image quality is the lowest. In terms of image quality, we can see improvement in the contrast level and a reduction in the entropy H_e applying the DCP algorithm (see Table. 3) in the SPI methods obtained, similar values contrast with variations in the entropy (the entropy affected the smoothing in the images) being the HSI method with H_e lowest.

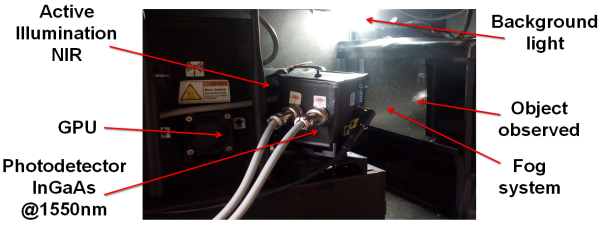


Figure 5: Experimental setup to vision system 3D NIR-SPI, the testing bench for simulate scattering environment [7].

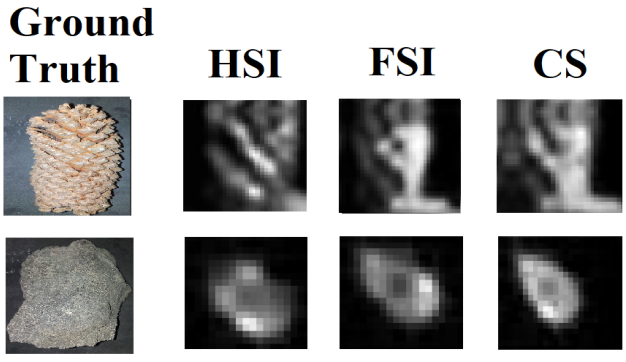


Figure 6: Reconstruction 2D NIR-SPI a scenario of foggy and midday using HSI, FSI, and CS methods.

http://www.imavs.org/

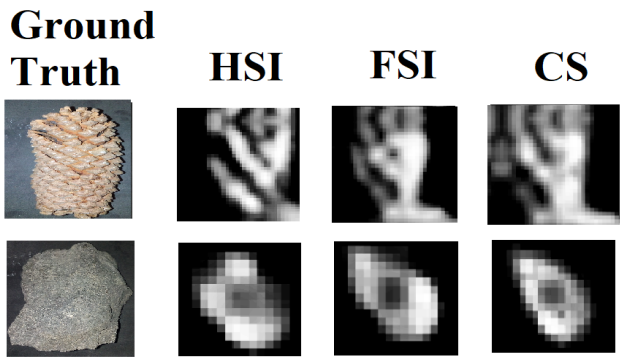


Figure 7: Reconstruction 2D NIR-SPI a scenario of foggy and midday using HSI, FSI, and CS methods applying DCP algorithm.

Table 1: EPI noise suppression applying DCP for the case a scenario with a foggy and background illumination of 15 KLux (half-cloudy)- 30 KLux (midday).

SPI method	EPI_{30KLux}	EPI_{15KLux}
CS	0.65	0.8
FSI	0.47	0.74
HSI	0.51	0.76

6 NIR SPI 2D IMAGE RECONSTRUCTION APPLIED TO UAVS AUTONOMOUS NAVIGATION IN OUTDOOR CONDITIONS

A vision system that operates in fog conditions is an advantage in Unmanned Aerial Vehicle (UAV) navigation applications. It provides fast deployment at remote or challenging locations, being essential for many civil and military applications such as post-disaster relief assistance [2]. Our NIR-SPI system developed, if adopted in UAVs (see Fig. 9), offers a solution to perform better visualization of the scene in outdoor conduction, if compared to conventional vision systems that normally use RGB sensors operating in the visible spectrum [7]. As explained, conventional systems operating in the VIS range present much higher image degradation in fog conditions than those operating in the NIR part of the spectra [7]. In the literature, there are examples of the use of IR vision systems to UAVs for applications such as rescue missions that enable fast exploration in scenes with low-illumination, or operations performed in complex weather conditions that limit the drone visibility. NIR-SPI system presented has great potential to be used in similar applications. Furthermore, SPI could be an alternative to the use of LIDAR technology.

7 CONCLUSIONS

This paper presented an evaluation of the HSI, FSI, and CS methods in a combination with the DCP algorithm as a strategy to improve image quality in a scattering environment

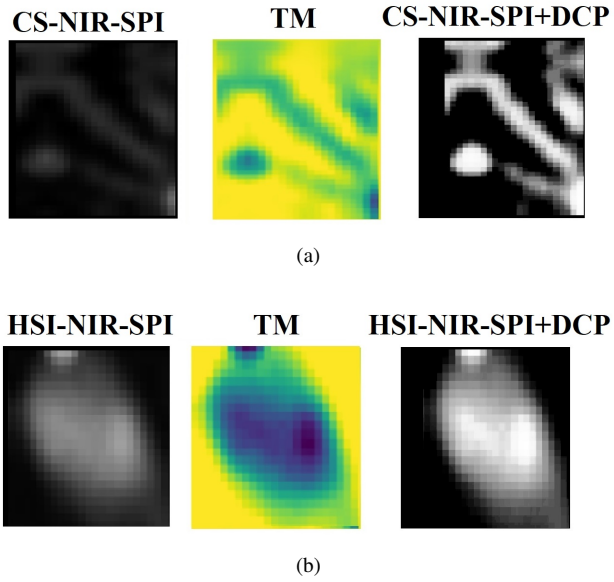


Figure 8: Reconstruction 2D NIR-SPI a scenario of foggy and half-cloudy using HSI and CS, in this scenario the 2D image NIR-SPI present low-contrast, obtained the TM and applying the DCP algorithm: a) pine seeds, and b) rock.

Table 2: Processing time 2D NIR-SPI DCP for the case a scenario with a foggy and background illumination of 15 KLux (half-cloudy)- 30 KLux (midday).

SPI method	$T_{30KLux}(ms)$	$T_{15KLux}(ms)$
CS	41	39
FSI	37	35
HSI	44	41

with different illuminations for a vision system SPC with active illumination for future use as a vision system in UFV. As test objects used rocks and pine seed elements in the forest environment. For the test, we generated foggy artificial and two illumination conditions half-cloudy and midday, in this simulated scenario presents a challenge to make good detection of the image, due to the attenuation effect caused by interacting the light with water particles. To improve the SPI image, we applied a DCP algorithm to the reconstructed image. In the case of the FSI, the SPI image reconstructed is more sensitive to source noise, an effect that we can observe in the evaluation of the term EPI (see Table. 1), and contrast (see Table. 3). For which is not recommended in SPI vision systems under scattering conditions. In the test, HSI and CS methods were shown to be robust under scattering conditions (see Table. 1) , mainly in the case low-contrast of the SPI image (see Fig. 8), being ideal to use as SPI reconstruction methods applied in the SPC. This capture capacity of SPI images under the scattering scenario shows the feasibility of using a vision system based on the SPI principle for applications to vehicles

http://www.imavs.org/

Table 3: Improvement the 2D NIR-SPI in terms entropy H_e, and contrast (C) applying DCP algorithm.

SPI method	H_e	C	H_{eDCP}	C_{DCP}
CS	7.39	10	6.3	11
FSI	7	9	5.3	9.8
HSI	6.74	9.95	5.2	10

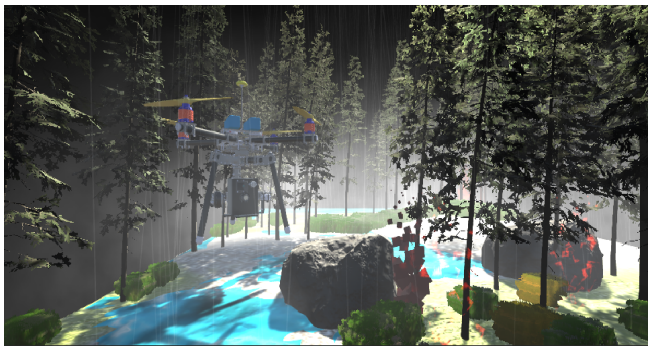


Figure 9: Schematic view to illustrate UAV (drone) navigation in presence of fog.

UFV as a redundant sensor vision for the RGB cameras.

ACKNOWLEDGEMENTS

The first author is thankful to Consejo Nacional de Ciencia y Tecnología (CONACYT) for his scholarship with No. CVU: 661331.

REFERENCES

[1] Ioannis Pitas. Drone vision and deep learning for infrastructure inspection. In *2021 IEEE International Conference on Autonomous Systems (ICAS)*, pages 1–1, 2021.

[2] Yogianandh Naidoo, Riaan Stopforth, and Glen Bright. Development of an uav for search amp; rescue applications. In *IEEE Africon '11*, pages 1–6, 2011.

[3] Zainab Zaheer, Atiya Usmani, Ekram Khan, and Mohammed A. Qadeer. Aerial surveillance system using uav. In *2016 Thirteenth International Conference on Wireless and Optical Communications Networks (WOCN)*, pages 1–7, 2016.

[4] Zhongping Lee and Shaoling Shang. Visibility: How applicable is the century-old koschmieder model? *Journal of the Atmospheric Sciences*, 73, 09 2016.

[5] Nathan Hagen. The influence of natural and artificial fogs on visible and infrared imaging. In Marija Strojnik and Maureen S. Kirk, editors, *Infrared Remote Sensing and Instrumentation XXVI*, volume 10765, pages 13 – 19. International Society for Optics and Photonics, SPIE, 2018.

[6] Hugh R. Carlon, David H. Anderson, Merrill E. Milham, Theodore L. Tarnove, Robert H. Frickel, and I. Sindoni. Infrared extinction spectra of some common liquid aerosols. *Appl. Opt.*, 16(6):1598–1605, Jun 1977.

[7] C. Osorio Quero, D. Durini, J. Rangel-Magdaleno, J. Martinez-Carranza, and R. Ramos-Garcia. Single-pixel near-infrared 3d image reconstruction in outdoor conditions. *Micromachines*, 13(5), 2022.

[8] Carlos A. Osorio Quero, Daniel Durini, Jose Rangel-Magdaleno, and Jose Martinez-Carranza. Single-pixel imaging: An overview of different methods to be used for 3d space reconstruction in harsh environments. *Review of Scientific Instruments*, 92(11):111501, 2021.

[9] Kaiming He, Jian Sun, and Xiaoou Tang. Single image haze removal using dark channel prior. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1956–1963, 2009.

[10] Xin Yang, Yang Liu, Xinyue Mou, Tianyu Hu, Fei Yuan, and En Cheng. Imaging in turbid water based on a hadamard single-pixel imaging system. *Opt. Express*, 29(8):12010–12023, Apr 2021.

[11] Graham M. Gibson, Steven D. Johnson, and Miles J. Padgett. Single-pixel imaging 12 years on: a review. *Opt. Express*, 28(19):28190–28208, Sep 2020.

[12] Zibang Zhang, Xueying Wang, Guoan Zheng, and Jingang Zhong. Hadamard single-pixel imaging versus fourier single-pixel imaging. *Opt. Express*, 25(16):19619–19639, Aug 2017.

[13] Giorgia Franchini, Roberto Cavicchioli, and Jia Cheng Hu. Stochastic floyd-steinberg dithering on gpu: image quality and processing time improved. In *2019 Fifth International Conference on Image Information Processing (ICIIP)*, pages 1–6, 2019.

[14] C. Osorio Quero, D. Durini, R. Ramos-Garcia, J. Rangel-Magdaleno, and J. Martinez-Carranza. Hardware parallel architecture proposed to accelerate the orthogonal matching pursuit compressive sensing reconstruction. In Lei Tian, Jonathan C. Petruccelli, and Chrysanthe Preza, editors, *Computational Imaging V*, volume 11396, pages 56 – 63. International Society for Optics and Photonics, SPIE, 2020.

[15] Wang Lin, Bi Du-Yan, Li Quan-He, and He Lin-Yuan. Single image dehazing based on visual-physical model. In Yu-Jin Zhang, editor, *Image and Graphics*, pages 360–368, Cham, 2015. Springer International Publishing.

http://www.imavs.org/

Handling Pitch Variations for Visual Perception in MAVs: Synthetic Augmentation and State Fusion

E. Cereda[1]*, D. Palossi[1,2], A. Giusti[1]
[1]Dalle Molle Institute for Artificial Intelligence (IDSIA) – USI and SUPSI – Lugano, Switzerland
[2]Integrated Systems Laboratory (IIS) – ETH – Zürich, Switzerland

http://www.imavs.org/

ABSTRACT

Variations in the pitch of a Micro Aerial Vehicle affect the geometry of the images acquired by its on-board cameras. We propose and evaluate two orthogonal approaches to handle this source of variability, in the context of visual perception using Convolutional Neural Networks. The first is a training data augmentation method that generates synthetic images simulating a different pitch than the one at which the original training image was acquired; the second is a neural network architecture that takes the drone's estimated pitch as an auxiliary input. Real-robot quantitative experiments tackle the task of visually estimating the pose of a human from a nearby nano-quadrotor; in this context, the two proposed approaches yield significant performance improvements, up to +0.15 in the R^2 regression score when applied together.

1 INTRODUCTION

In autonomous MAVs, convolutional neural networks (CNNs) are frequently adopted to process on-board camera images to solve perception problems [1, 2, 3, 4]. This is especially true in nano-sized MAVs (i.e., sub-10cm span and few tens of grams weight), where tight computational and memory resources highly constrain the affordable methods.

To achieve consistent performance, these CNNs must be robust to sources of variability that may occur during deployment. One such example is the continuously-changing orientation of the camera, caused in quadrotors by pitch and roll changes required to generate accelerations; while larger drones feature mechanically, optically or electronically stabilized cameras, this is not the case for nano-quadrotors.

In this work, we consider the task of human pose estimation, leveraging the PULP-Frontnet CNN [4] to visually estimate the relative pose of a person from a low-resolution image acquired by a nano-quadrotor flying in its proximity. To ensure good performance in the field, this model should be as insensitive as possible to changes in drone attitude.

As a **main contribution**, we propose an ad-hoc training data augmentation method that synthesizes images as if acquired from a different pitch than the one at which the original image was acquired; this improves robustness to drone pitch changes with no impact on inference time; the approach is therefore well suited for deployment on nano-quadrotors.

As a **secondary contribution**, we propose to provide the camera pitch angle as an auxiliary input to the CNN, to facilitate its perception task. In fact, a good pitch estimate is always available from the quadrotor's inertial measurement unit (IMU) and state estimation subsystem; because pitch affects the observed image geometry, we hypothesize that its knowledge helps the model to correctly interpret the image data. Similarly, neuroscience research [5, 6] found that information from the vestibular system, which encodes the orientation of the head, contributes to the human visual perception.

After reviewing related work (Section 2), our contributions are described in Section 3. In Section 4 we describe our experimental setup; in Section 5, quantitative experiments demonstrate significant benefits of both our proposed contributions.

2 RELATED WORK

Camera images, like most high-dimensional data, often allow for large-scale variations which nonetheless have no impact on the image's semantic content, depending on the task being considered. For example, illumination or image exposure changes should not affect an object classification pipeline. To build systems invariant to such semantically-irrelevant changes, two high-level approaches are commonly used: *i)* normalization and *ii)* data augmentation.

Normalization eliminates a source of variation altogether by bringing input images to a single canonical representation. For illumination, this corresponds to equalization algorithms such as CLAHE [7]; to compensate camera movements, digital, optical, and mechanical stabilization methods exist [8]. Image stabilization has been widely deployed in computer vision, including in MAV applications: e.g. the Parrot Bebop 2 uses digital stabilization, whereas the DJI Phantom 2 relies on a mechanical gimbal. Its main drawback is cost at runtime, whether in computational power or in mechanical complexity, which, to this date, prevents adoption on nano-sized MAVs.

Alternatively, CNNs can learn arbitrary invariants when they are sufficiently represented in the training data. As real-world data is typically scarce, especially in robot applications, deep learning pipelines make extensive use of data augmentation for this purpose: available data is artificially in-

*Email address(es): elia.cereda@idsia.ch

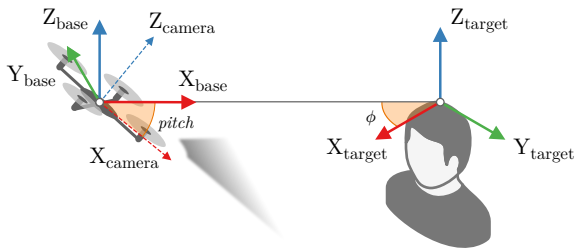


Figure 1: Human pose estimation task and reference frames.

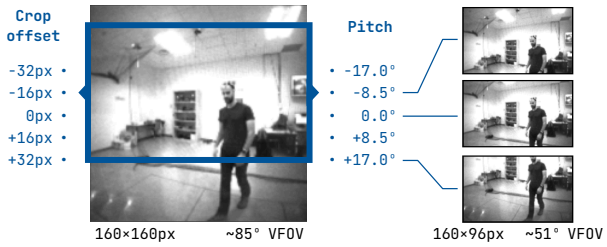


Figure 2: Pitch augmentation technique and example images.

creased by randomly perturbing copies of each training instance with transformations that do not affect the corresponding target variables. Being applied only at training time, augmentation is ideal for applications with tight resource constraints at inference time, such as nano-drones.

A rich data augmentation literature exists [9]. Generic photometric adjustments that simulate illumination changes are some of the most widely applied, together with small-scale geometric perturbations [10]. Task-specific augmentation strategies have also been proposed: *domain randomization* to improve generalization to unseen environments [11, 12]; *view synthesis* to increase the density of camera poses for visual localization and semantic segmentation [13, 14]; meta-learning approaches allow automatic task-specific fine-tuning of augmentations [15]. Our proposed *pitch augmentation* technique can be seen as a constrained version of view synthesis, which deals exclusively with camera orientation changes along the pitch axis.

Directly-measurable sources of variations allow an additional, orthogonal strategy: the current state can be fed as an explicit additional input to the CNN (vision-state fusion). In robotic control-oriented tasks and in egocentric spatial perception tasks, where the output refers to the system itself, this is an established practice. Abbeel et al. [16] use a drone's linear and angular velocities as the only inputs of a learned model that performs acrobatic maneuvers; more recent control approaches integrate the state and visual features in a single neural network [17]. Egocentric visual-inertial state estimation uses camera images and the robot's velocities and orientations measured by an inertial measurement unit (IMU), either through a sequence-to-sequence recurrent neural network that fuses the two input sequences and produces the sequence of egocentric poses [18, 19], or through a feed-forward architecture that takes an inertial initial motion estimation and refines it using the visual feed [20].

In this work, instead, we consider human relative pose estimation: an *allocentric* spatial perception task, whose output is external to the robot. Current CNN-based approaches for this class of problems rely exclusively on visual inputs [4, 21] but, for them too, recent advances show benefits when exploiting additional state information [22]. We evaluate a modified CNN architecture that receives the current pitch as additional input from the drone's onboard state estimation.

3 Method

3.1 Application

The reference application for our work is human pose estimation aboard the Crazyflie 2.1[1] 27-gram MAV by Bitcraze, coupled with the AI-deck pluggable extension board which provides a Himax HM01B0 gray-scale QVGA camera and a GreenWaves Technologies GAP8 ultra-low power multi-core system-on-chip (SoC). We leverage the *PULP-Frontnet* convolutional neural network, a field-proven model which takes one gray-scale 160×96px image and estimates the pose of the human subject relative to the drone (see Figure 1). The model produces four independent regression outputs, which represent the Cartesian coordinates of the subject's position in 3D space (x, y, z) w.r.t. the drone's horizontal $\{\text{base}\}$ frame and the subject's relative orientation w.r.t. the drone's yaw, ϕ. In this work, we focus on the position components (x, y, z).

To train our models, we employ ground-truth data collected in rooms equipped with a motion capture (mocap) system. Images streamed from the drone's camera are stored together with the corresponding drone's and subject's absolute poses, as recorded by the mocap system, and the drone's attitude, computed by onboard state estimation.

3.2 Pitch augmentation

During flight, the quadrotor maneuvers by continuously adjusting its pitch to generate forward/backward thrust. Without any kind of digital or mechanical stabilization (i.e., normalization), these attitude changes directly affect the geometry of the drone camera images. Pitch augmentation addresses this problem by synthetically enhancing the pitch range represented in the CNN's training set, at the expense of a reduced vertical field of view (VFOV). As shown in Figure 2, we acquire 160×160px images from the Himax camera with a $85°$ VFOV. By cropping 160×96px sub-regions at varying vertical offsets, we can simulate 65 different images with a $51°$ VFOV in a $\pm 17°$ range of relative pitch compared to the original image. In other words, the top 96-row region of an image shows approximately the same scene at a $-17°$ pitch compared to the central 96-row region.

This simple approximation neglects the perspective and radial distortion components of the real geometric transfor-

[1] https://bitcraze.io/products/crazyflie-2-1/

http://www.imavs.org/

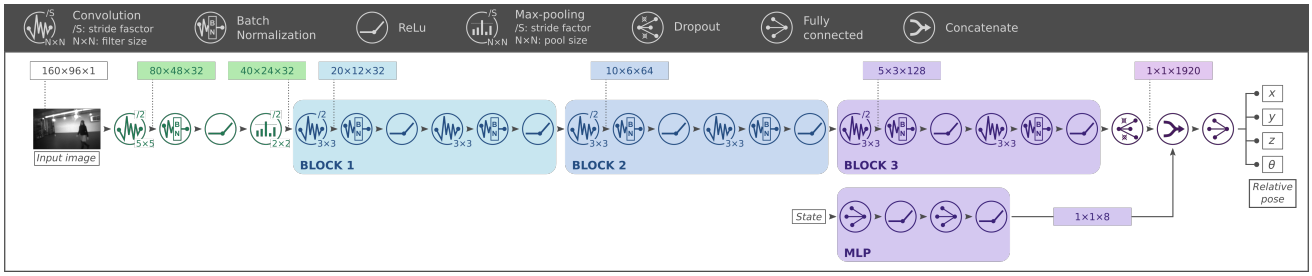


Figure 3: PULP-Frontnet CNN architecture [4], extended with the multi-layer perceptron branch for the state input.

mation imposed by a pitch change, but has the advantage of generating augmented sub-regions that are always entirely contained in the original image.

3.3 *Vision-state fusion*

We also propose an orthogonal approach to deal with changes in drone attitude by feeding to the model the current pitch in addition to the image; we expect that the CNN will exploit this information to explicitly account for pitch changes. This takes advantage of information already available on board, computed by the state estimation system from measurements of the inertial measurement unit (IMU).

We compare the four models that result from combining pitch augmentation applied to the training data (referred from here on as models *w/* or *w/o pitch aug*) and the additional pitch input fed to the CNN (i.e., *stateless* and *stateful* models). When evaluating models trained with both pitch augmentation and stateful input, care must be taken to ensure that the state input remains consistent after the augmentation: we sum the original image's ground-truth pitch with the synthetic pitch angle that corresponds to the random region selected by the augmentation. The resulting angle, which is then fed to the model, is the pitch that would produce a real image with the same orientation as the augmented one.

4 Experimental setup

Our model architecture is based on PULP-Frontnet [4], a 9-layer CNN tuned for the GAP8 SoC on the target nano-drone. We follow the original execution strategy when deploying the trained models: 8-bit quantized arithmetic for all CNN operations and automatically-generated tiling code, respectively to compensate for the lack of floating-point units on the GAP8 and to efficiently exploit its explicitly-managed memory hierarchy and 8-core parallel cluster.

For our vision-state fusion experiments, we extend our architecture to receive the drone's current pitch from the state estimation as an additional scalar input. As described in [22], we pre-process the state input with a small multi-layer perceptron (MLP), composed of two 8-unit fully connected layers (FC) interleaved by ReLU non-linearities, before concatenating it to the CNN's main FC layer. The resulting architecture (see Figure 3) has a negligible increase in inference-time memory and computation requirements, respectively +120 bytes and +140 multiply-accumulate operations (MACs), compared to 300 kB and 14 MMACs total for the original PULP-Frontnet architecture. We deploy the MLP branch with sequential software-emulated floating-point arithmetic, which accounts for just 0.5% of the total workload ($\sim$ 20k clock cycles).

Data for our experiments is collected in two different mocap-equipped indoor laboratories and includes a combination of static samples with the drone fixed horizontally on a wheeled cart (20%) and in-flight samples with the drone controlled by a human pilot (80%). In total, we record 12k camera frames, with associated ground-truth poses and estimated attitudes, from sessions with 17 human subjects of different age, height, ethnicity, and clothing. Three subjects (4.7k samples) form our test set, while the remaining 14 subjects (7.3k) are split into training (90%) and validation (10%) sets.

In our strategy, we augment each training sample 10 times, as an offline pre-processing step, performing a number of standard photometric data augmentations to promote robustness of our model to illumination changes – exposure, gamma correction, dynamic range adjustments, addition of Gaussian noise, and blurring – followed by vignetting, and horizontal flipping with 50% probability (shown in Figure 4). When enabled, our proposed pitch augmentation is also applied during this step, before all other augmentations. On the training and validation sets, we uniformly select a random synthetic pitch from the $\pm 17°$ range. For one experiment, we perform pitch augmentation also at test time – without the other augmentations – in this case generating every possible synthetic pitch for each test image (65 copies). Finally, we discard samples in which the subject is outside the field of view, obtaining the distributions of camera pitch values shown in Figure 5. We observe that pitch augmentation has a strong regularizing effect on the distribution, removing the heavy bias towards pitch $0°$ of the real data – i.e., w/o pitch aug in Figure 5.

Training is performed for 100 epochs using the Adam optimizer with learning rate 10^{-3} to minimize the L1 loss of the relative pose. At the end of training, the model checkpoint that reached the best performance on the validation set is selected for evaluation on the test set.

http://www.imavs.org/



Figure 4: Individual photometric data augmentations (top). Ten images produced by the entire augmentation pipeline (bottom).

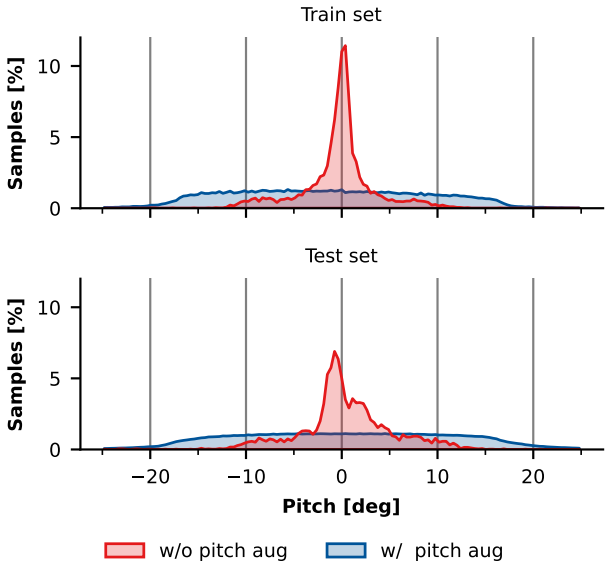


Figure 5: Distributions of camera pitch in the datasets.

5 Experimental Results

In the following, we first evaluate the performance of our models in offline experiments on the test set. We train five instances of each model and measure their regression performance using the coefficient of determination R^2, a standard adimensional metric that represents the fraction of variance in the target variable explained by the model[2]. An $R^2 = 1.0$ corresponds to a perfect regressor, while a dummy regressor that always outputs the mean of the test data achieves $R^2 = 0.0$. Models can perform arbitrarily worse than the dummy regressor, leading to negative R^2 scores. The R^2 score is closely related to another standard regression metric, the mean squared error (MSE)[3]. Unlike the MSE, the R^2 score quantifies the quality of the regressor independently of the variance of the target variable, and is therefore suitable for comparing regression performance on different variables. To further validate our results, we also evaluate the entire closed-loop system, deploying the models aboard the nano-drone and evaluate their in-flight behavior.

5.1 *Offline experiments*

We show the four models' regression performance in terms of R^2 score on the test set in Figure 6-A. The *stateless* model *w/o pitch aug* (hollow red model) represents our baseline. Introducing pitch augmentation alone (hollow blue) corresponds to the original PULP-Frontnet model, trained and evaluated on our new data. We see a strong positive effect on the performance of all three output components (x, y, z) when compared to the baseline, with an increase in median R^2 of respectively $+0.13$, $+0.12$, and $+0.09$. The original PULP-Frontnet is trained exclusively on static training data at $0°$ pitch, with pitch augmentation as the only source of pitch variability. In contrast, we show in this work that pitch augmentation is strongly beneficial even when a vast majority of training data is collected in flight with real pitch variations.

On the other hand, introducing the state input alone (filled red) has a weak effect compared to the baseline. Only on z we see a median R^2 increase of $+0.07$, while x and y are much closer ($+0.02$ and $+0.00$). A similarly weak effect is visible when the state input is introduced on top of pitch augmentation (filled blue vs. hollow blue): on z we see a median R^2 increase of $+0.06$, while x and y are not affected. Overall, pitch augmentation strongly improves all three output components, while the state input noticeably benefits only z (intuitively, the component on which pitch has the strongest visual impact). The two techniques can be applied simultaneously, with the respective benefits compounding to reach the top performance.

We further analyze the four models' regression performance across a wide range of pitch angles by applying test-time pitch augmentation to the test set, as discussed in Section 4. Unsurprisingly, the overall regression performance in Figure 6-B decreases for all models compared to Figure 6-A. Those trained without pitch augmentation (red) in particular show a drop in median R^2 score of up to -0.33. Relative performance across the four model is unchanged compared to the raw test set. The nearly-uniform distribution imposed

[2]Defined as $R^2 = 1 - \sum_i (y_i - \hat{y}_i)^2 / \sum_i (y_i - \bar{y})^2$ with y_i the ground-truth output and $\hat{y}_i$ the model prediction for each test sample i, $\bar{y}$ the mean of ground-truth outputs.

[3]In fact, rewriting the formula above gives $R^2 = 1 - \text{MSE}/\text{MSE}_{dummy}$.

http://www.imavs.org/

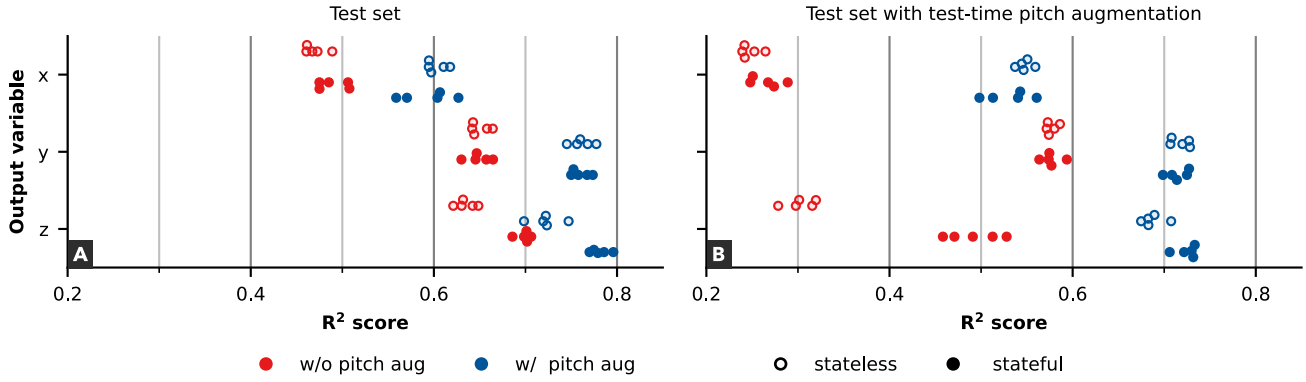


Figure 6: Overall regression performance in the offline experiments on the test set.

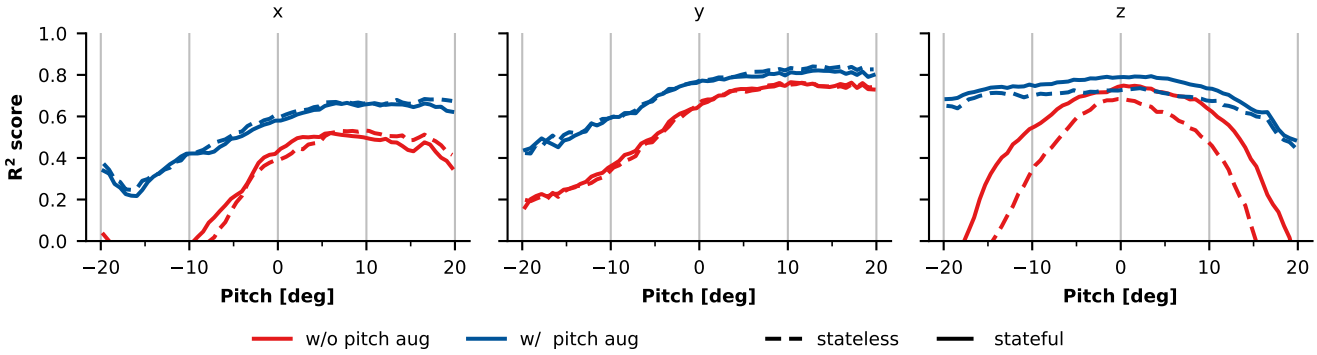


Figure 7: Regression performance on the test set with test-time pitch augmentation, broken down by drone pitch.

by pitch augmentation allows us to clearly visualize the regression performance broken down by pitch angle in Figure 7. Non-augmented models (red) exhibit a drastic drop at extreme pitch angles, especially on z, where performance falls to $R^2 = 0.0$ at $\pm 15°$. The performance of augmented models (blue) remains more stable across the entire range of pitch values: we attribute this to scarcity of real-world training data at extreme pitches, further confirming the effectiveness of pitch augmentation. On top of that, performance on x and y decreases at negative pitch angles (camera looking up), with pitch augmentation only partially compensating. Often, from visual inspection of the data, just the subject's head remains visible at strongly negative pitch, making accurate predictions harder.

5.2 *In-field evaluation*

In the second experiment, we focus on the two best-performing models, those trained with pitch augmentation (blue), and evaluate their behavior in the field when deployed aboard a closed-loop fully-autonomous Crazyflie nano-drone. We evaluate a combination of two scenarios and three distinct subjects, repeating each for five flights of $\sim$10 s per model, a total of 60 test flights. In both scenarios, the drone starts hovering at 0.5 m altitude and at a $\sim$6 m horizontal distance in front of the subject, then the onboard CNN inference is used to autonomously control the drone towards a target position 1.3 m in front of the person at eye level. Forward velocity is limited to a maximum of 1.2 m/s.

In the first scenario, the subjects stand still for the entire duration of the test, while the drone climbs from the hovering altitude up to their height (1.65–1.85 m) and simultaneously moves forward towards them. We are thus stressing the models' x and z predictions. Figure 8-A shows the drone's distance from the target in the horizontal plane over time under this scenario: both models achieve almost identically good behavior, converging to the target with almost no error nor oscillations. Figure 8-B, on the other hand, shows the difference (delta) between drone and target altitudes over time. Here, the models' behaviors differ noticeably: while the stateful model firmly converges to the desired position, the stateless model cannot reach it and overshoots instead.

The second scenario is identical to the first at the beginning. Then, after 6 s from the start, the subjects are instructed to kneel down. This strongly challenges dynamic z predictions, as the drone has almost reached the target position 1.3 m in front of the subjects when they start kneeling.

http://www.imavs.org/

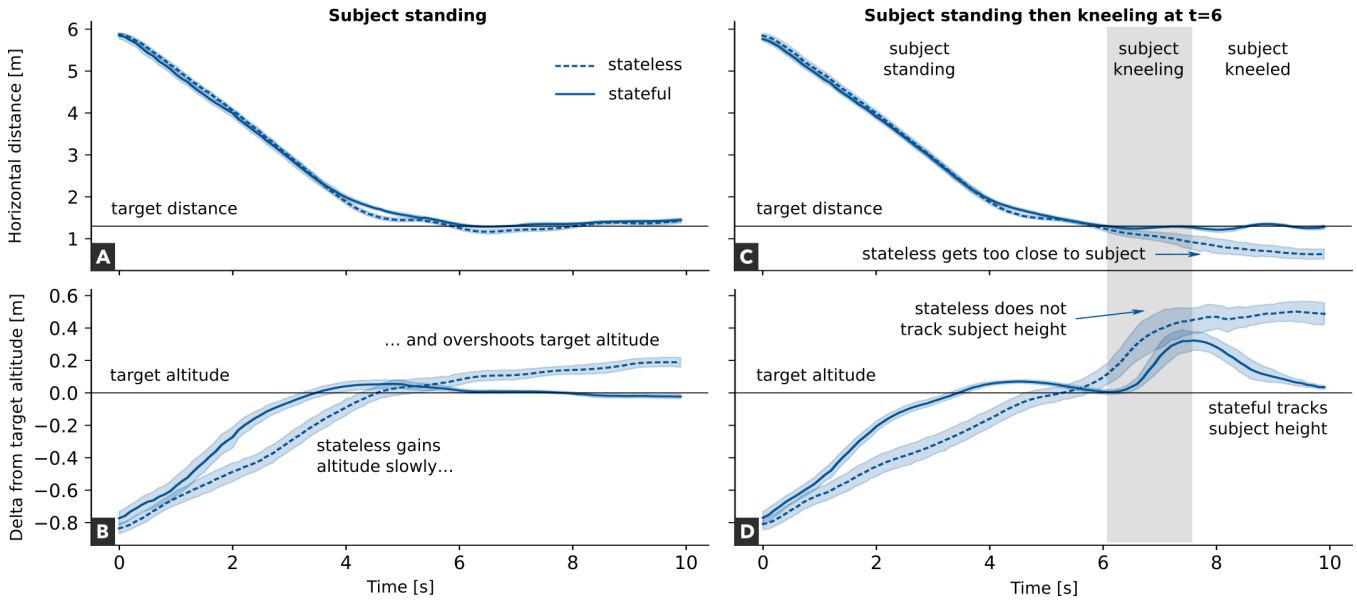


Figure 8: In-field behavior of the stateless and stateful models trained with pitch augmentation.

This means the person's downward motion corresponds to a considerable movement in image space, giving the drone little time to react before losing the target outside the field of view. Under this scenario, we see the stateless model struggling to follow subjects as they kneel, both in the horizontal plane (Figure 8-C) and in altitude (Figure 8-D). In contrast, the stateful model both precisely converges in front of the standing subjects ($t = 6$s), and accurately follows them as they kneel ($t = 6$s to 7.5s), keeping the correct horizontal distance and descending to reach the new target altitude.

From this experiment, we conclude that the state input (pitch) leads to significantly improved performance in the field. While little difference between the stateless and stateful models was seen in the offline experiments, we show that the stateful model's in-flight closed-loop behavior significantly outperforms its stateless counterpart.

6 Conclusion

We considered a perception problem for MAVs solved using a regression CNN; to enforce robustness to image variations due to drone pitch, we proposed and quantitatively evaluated an approach based on data augmentation, achieving improvements in median R^2 score of up to +0.15; we further verified that providing explicit information about the drone pitch to the model yields improvements both in offline and in-field control experiments.

Acknowledgements

This work was partially supported by the Secure Systems Research Center (SSRC) of the UAE Technology Innovation Institute (TII).

References

[1] Guanya Shi, Wolfgang Hönig, Yisong Yue, and Soon-Jo Chung. Neural-Swarm: Decentralized close-proximity multirotor control using learned interactions. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3241–3247, 2020.

[2] Wenda Zhao, Jacopo Panerati, and Angela P. Schoellig. Learning-based bias correction for time difference of arrival ultra-wideband localization of resource-constrained mobile robots. *IEEE Robotics and Automation Letters*, 6(2):3639–3646, 2021.

[3] Daniele Palossi, Francesco Conti, and Luca Benini. An open source and open hardware deep learning-powered visual navigation engine for autonomous nano-UAVs. In *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 604–611. IEEE, 2019.

[4] Daniele Palossi, Nicky Zimmerman, Alessio Burrello, Francesco Conti, Hanna Müller, Luca Maria Gambardella, Luca Benini, Alessandro Giusti, and Jérôme Guzzi. Fully onboard AI-powered human-drone pose estimation on ultra-low power autonomous flying nano-UAVs. *IEEE Internet of Things Journal*, 9(3):1913–1929, 2021.

[5] Naotoshi Abekawa, Elisa R Ferrè, Maria Gallagher, Hiroaki Gomi, and Patrick Haggard. Disentangling the visual, motor and representational effects of vestibular input. *Cortex*, 104:46–57, 2018.

http://www.imavs.org/

[6] Elisa R Ferrè, Adrian JT Alsmith, Patrick Haggard, and Matthew R Longo. The vestibular system modulates the contributions of head and torso to egocentric spatial judgements. *Experimental Brain Research*, 239(7):2295–2302, 2021.

[7] Stephen M. Pizer, E. Philip Amburn, John D. Austin, Robert Cromartie, Ari Geselowitz, Trey Greer, Bart ter Haar Romeny, John B. Zimmerman, and Karel Zuiderveld. Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing*, 39(3):355–368, 1987.

[8] Marcos Roberto e Souza, Helena de Almeida Maia, and Helio Pedrini. Survey on digital video stabilization: Concepts, methods, and challenges. *ACM Comput. Surv.*, 55(3), feb 2022.

[9] Connor Shorten and Taghi Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6, 07 2019.

[10] Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems*, volume 33, pages 6256–6268. Curran Associates, Inc., 2020.

[11] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[12] Elia Cereda, Marco Ferri, Dario Mantegazza, Nicky Zimmerman, Luca M. Gambardella, Jérôme Guzzi, Alessandro Giusti, and Daniele Palossi. Improving the generalization capability of DNNs for ultra-low power autonomous nano-UAVs. In *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 327–334, 2021.

[13] Yiming Wan, Wei Gao, Sheng Han, and Yihong Wu. Boosting image-based localization via randomly geometric data augmentation. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 688–692, 2020.

[14] Joris Guerry, Alexandre Boulch, Bertrand Le Saux, Julien Moras, Aurelien Plyer, and David Filliat. SnapNet-R: Consistent 3D multi-view semantic labeling for robotics. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2017.

[15] Barret Zoph, Ekin D Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V Le. Learning data augmentation strategies for object detection. In *European Conference on Computer Vision*, pages 566–583. Springer, 2020.

[16] Pieter Abbeel, Adam Coates, and Andrew Y Ng. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, 29(13):1608–1639, 2010.

[17] Elia Kaufmann, Antonio Loquercio, Rene Ranftl, Matthias Mueller, Vladlen Koltun, and Davide Scaramuzza. Deep drone acrobatics. In *ROBOTICS: SCIENCE AND SYSTEMS XVI*, 2020.

[18] Ronald Clark, Sen Wang, Hongkai Wen, Andrew Markham, and Niki Trigoni. VINet: Visual-inertial odometry as a sequence-to-sequence learning problem. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), Feb. 2017.

[19] Liming Han, Yimin Lin, Guoguang Du, and Shiguo Lian. DeepVIO: Self-supervised deep learning of monocular visual inertial odometry using 3d geometric constraints. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6906–6913, 2019.

[20] Sudeep Pillai and John J. Leonard. Towards visual ego-motion learning in robots. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5533–5540, 2017.

[21] Elia Kaufmann, Mathias Gehrig, Philipp Foehn, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Beauty and the beast: Optimal methods meet learning for drone racing. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 690–696. IEEE, 2019.

[22] Elia Cereda, Stefano Bonato, Mirko Nava, Alessandro Giusti, and Daniele Palossi. Vision-state fusion: Improving deep neural networks for autonomous robotics, 2022. [online pre-print].

http://www.imavs.org/

Using trajectory oscillation timing improves in-flight odometry based solely on optic flows

L. Bergantin[1] *, C. Coquet[1], A. Negre[2], T. Raharijaona[3], N. Marchand[2] and F. Ruffier[1]
[1] Aix-Marseille Université, CNRS, ISM, Marseille, 13009 Marseille, France
[2] Gipsa-Lab, CNRS, Université Grenoble Alpes, 38402 Saint-Martin-d'Hères, France
[3] Université de Lorraine, Arts et Métiers Institute of Technology, LCFC, HESAM Université, F-57070 Metz, France

http://www.imavs.org/

ABSTRACT

Traveled distance estimation is a common problem for robotic applications taking place in unknown environments where GPS is not available. In drones, the presence of weight and computational power constraints leads to the importance of developing odometry strategies based on minimilastic equipment. In this study, we imposed upon a hexarotor to perform up-and-down oscillatory motions while flying forward to test a self-scaled scheme of a visual odometer for the first time. For the odometry, the downward translational optic flow was scaled by the current visually estimated flight height and then mathematically integrated to evaluate the total distance traveled. The self-oscillatory trajectory generated successions of contraction and expansion in the optic flow vector field, which allowed to estimate the flight height of the hexarotor by means of an Extended Kalman Filter. We present three strategies based on sensor fusion that rely on no, precise or rough prior knowledge of the optic flow variations imposed by the sinusoidal trajectory. The rough prior knowledge strategy uses solely the timing of the variations of the optic flow. Tests were performed in a flying arena, where the hexarotor followed a circular trajectory while oscillating up-and-down over about $50m$ under illuminances of $117lux$ and $1518lux$.

1 INTRODUCTION

Traveled distance estimation of an aerial robot in an unknown environment is a common problem for all types of applications when GPS is not available. In drones, the need to reduce the Size, Weight and Power (SWaP) of the perception equipment is often of great importance to ensure the success of the task.

Several visual odometric approaches involving the use of either optic flow [1, 2], events, images & IMU combination [3] or the sparse-snapshot method [4] have been successfully tested on flying robots. All these approaches require ground height information providing the factor scaling the visual information. This scaling factor can be determined separately using a static pressure sensor [5] or stereovision [1, 4] or is integrated in the hybrid approach [3], for example. One solution to estimate the 2D position of a drone in an unknown environment is concurrent onboard odometry and visual mapping, as well as onboard SLAM (Simultaneous Localisation and Mapping) [6, 7, 8]. A minimalistic alternative is IMU (Inertial Measurement Unit) based dead reckoning - i.e. inertial integration [9]. The dead reckoning position signal could be used by a flying robot to get close enough to detect a landmark before reaching it, giving a new known starting point.

Most of these approaches require the use of computationally intensive algorithms and a feedback from the environment (such as the detection of a beacon or the feedback from a map). A minimalistic alternative is the use of optic flow cues, such as translational optic flow and optic flow divergence. Translational optic flow has been used on flying robots to visually control landing [10], to follow uneven terrain [11] and to attempt visual odometry and localisation [12, 13] (see [14] for review).

Self-oscillations have been observed in honeybees flying forward in both longitudinal and vertical tunnels ([15] and [16], respectively). The self-oscillatory motion generates a series of expansions and contractions in the optic flow vector field: the optic flow divergence cue. Visually controlled landing was achieved by using optic flow divergence [17, 18, 19, 20]. The instabilities due to the oscillatory movement have been used to determine the flight height of a micro-flyer by exploiting the linear relation between the oscillation and the fixed control gain [19]. The instabilities due to depth variation have been used to assess the optic flow scale factor of the observed scene to perform visual odometry onboard an underwater vehicle [21]. In [22], the local optic flow divergence was measured by means of two optic flow magnitudes perceived by two optic flow sensors placed on a chariot performing back-and-forth oscillatory movements in front of a moving panorama. The local optic flow divergence was then used to estimate the local distance between the chariot and the moving panorama by means of an Extended Kalman Filter (EKF).

In this study, we investigate how to include some knowledge about the trajectory oscillations in an odometry strategy based only on optic flow cues. The optic flow based odome-

*Email address: lucia.bergantin@univ-amu.fr

http://www.imavs.org/

try scheme, called SOFIa (**S**elf-scaled **O**ptic **F**low time-based **I**nteg**a**tion model), was tested here for the first time on a hexarotor equipped with optic flow sensors that oscillated up-and-down following a circular trajectory of about $50m$. The SOFIa model was previously assessed in bio-plausible simulations to model the visual odometer of honeybees [23]. The estimation of the distance traveled by means of the SOFIa method is based on the integration of the local translational optic flow scaled by the flight height of the drone, which is estimated by means of an EKF taking the local optic flow divergence as measurement. Such integration scheme can therefore be considered as a minimalistic dead reckoning solution based on optic flow.

First, we applied the SOFIa method using only 2 optic flow measurements perceived along the longitudinal axis of the drone, with no prior knowledge of the optic flow variations. Then, to increase the odometry accuracy, we tested a sensor fusion strategy based on the parameters of the self-oscillation using 4 optic flow sensors embedded on the hexarotor. The idea was to use some prior knowledge of the oscillations imposed upon the drone to better evaluate the optic flow divergence and the translational optic flow cues. We tested two different sensor fusions based respectively on a precise and on a rough prior knowledge of the optic flow variations. The sensor fusion based on a rough prior knowledge uses solely the timing of the variations of the optic flow. All three optic flow based odometry processing were tested on bouncing circular trajectories of about $50m$ under illuminances of $117lux$ and $1518lux$.

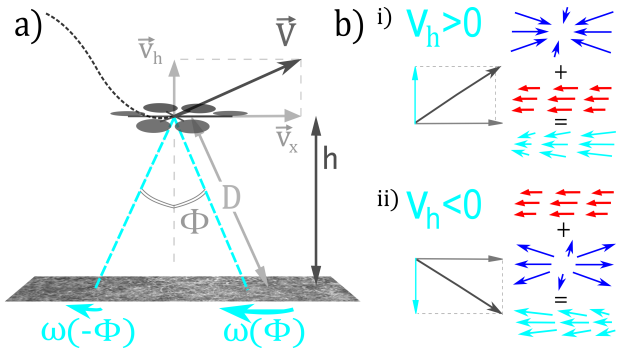


Figure 1: Hexarotor oscillating up-and-down while flying forward over the ground. a) The hexarotor's velocity V can be decomposed in the components V_x and V_h. Along the hexarotor's longitudinal axis x, the optic flow sensors are set at angles $\pm\phi$ with respect to the hexarotor's vertical axis and at a distance D with respect to the ground. They perceive the optic flow magnitudes $\omega(\phi)$ and $\omega(-\phi)$, respectively. This configuration is found along the hexarotor's lateral axis y as well. b) If V_h is positive, the optic flow divergence component is a contraction (i); if it is negative, the optic flow divergence component is an expansion (ii). The contraction or expansion of the optic flow is superimposed in the ventral optic flow vector field on the translational optic flow.

In section 2, we discuss the measurement of the local translational and divergence optic flow cues. In section 3, we discuss the minimalistic visual odometer method. In section 4, we describe the hexarotor and the optic flow sensors used. In section 5, we describe the odometry processing based on the raw measurements of 2 optic flow sensors without any prior knowledge of the optic flow variations. In section 6, we describe the sensor fusion odometry processing based on 4 optic flow sensors, both with a precise and with a rough prior knowledge of optic flow variations. In section 7, we show experimentally that the sensor fusion strategies based on the knowledge of optic flow variations increase the measurement accuracy of the local optic flow cues by comparing the three methods. Finally, we compare the performance of the minimalistic in-flight optic flow based odometry of the three methods. In section 8, conclusions are drawn and future works are discussed.

2 MEASUREMENT OF THE LOCAL OPTIC FLOW CUES

The translational optic flow is the pattern generated on the optic flow vector field by the translational motion of a drone flying above the ground [24]. The theoretical local translational optic flow ω_T^{th} can be expressed as the ratio between the V_x component of the drone's velocity and its flight height h (see Figure 1):

$$\omega_T^{th} = \frac{V_x}{h} \tag{1}$$

The local translational optic flow can be measured on a hexarotor as the sum of two optic flow magnitudes $\omega(\phi)$ and $\omega(-\phi)$ perceived by two optic flow sensors oriented at angles $\pm\phi$ with respect to the hexarotor's vertical axis, divided by a known factor of $2 \cdot \cos(\phi)^2$:

$$\omega_T^{meas} = \frac{\omega(\phi) + \omega(-\phi)}{2 \cdot \cos(\phi)^2} = \frac{V_x}{h} \tag{2}$$

In the case of a hexarotor equipped with four optic flow sensors as illustrated in Figure 2.b, three translational optic flow cues can be measured as:

- the sum of the two optic flow magnitudes perceived by the two optic flow sensors set along the longitudinal axis x, namely $\omega_{T_1}^{meas}$,
- the sum of the two optic flow magnitudes perceived on the x axis by the two optic flow sensors set along the lateral axis y, namely $\omega_{T_2}^{meas}$,
- the median of the four optic flow magnitudes considered, projected on the hexarotor's vertical axis by a $1/\cos(\phi)$ factor, namely $\omega_{T_3}^{meas}$.

The series of contractions and expansions generated in the optic flow vector field by up-and-down oscillatory motions is known as optic flow divergence. When a drone flies forward

while oscillating up-and-down above the ground, in the optic flow vector field the optic flow divergence is superimposed on the translational optic flow. Due to the oscillatory movement, the state vector $X = [h, V_h]^T$ is locally observable [20]. The theoretical local optic flow divergence ω_{div}^{th} can be expressed as the ratio between the V_h component of the drone's velocity and h (see Figure 1):

$$\omega_{div}^{th} = \frac{V_h}{h} \tag{3}$$

In [22], the authors have mathematically demonstrated that the local optic flow divergence can be measured on a micro-flyer as the subtraction between two optic flow magnitudes $\omega(\phi)$ and $\omega(-\phi)$ perceived by two optic flow sensors oriented at angles $\pm\phi$ with respect to the normal to a surface, divided by a known factor of $\sin(2\phi)$:

$$\omega_{div}^{meas} = \frac{\omega(\phi) - \omega(-\phi)}{\sin(2\phi)} = \frac{V_h}{h} \tag{4}$$

In the case of a hexarotor equipped with four optic flow sensors, two optic flow divergence cues can be measured as:

- the subtraction between the two optic flow magnitudes perceived by the two optic flow sensors set along the longitudinal axis x, namely $\omega_{div_x}^{meas}$,
- the subtraction between the two optic flow magnitudes perceived by the two optic flow sensors set along the lateral axis y, namely $\omega_{div_y}^{meas}$.

3 THE SOFIA VISUAL ODOMETER METHOD

In [23], the authors have assessed in simulation a model of the honeybee's visual odometer called SOFIa (**S**elf-scaled **O**ptic **F**low time-based **I**nteg**ra**tion model). The SOFIa model is based on the integration of the local translational optic flow ω_T scaled by the estimated distance with respect to the ground $\hat{h}$:

$$\hat{X}_{SOFIa} = \int \omega_T \cdot \hat{h}\, dt \tag{5}$$

In [23], $\hat{h}$ was estimated by means of an EKF taking as input the honeybee's wing stroke amplitude and as measurement the local optic flow divergence computed as the ratio between V_h and h (as in equation (3)). The local translational optic flow was computed as the ratio between V_x and h (as in equation (1)). The SOFIa model was found to be about 10 times more accurate than the raw mathematical integration of optic flow.

4 MATERIALS AND METHODS

The hexarotor was developed together with Hexadrone™ and equipped with four Pixart PAW3903 optic flow sensors (see Figure 2 and Table 1). The Pixart PAW3903 optic flow sensors were embedded on printed circuits to set them on the drone. The hexarotor had as onboard low-level flight controller the PX4 autopilot system [25] and used a trajectory tracking algorithm[1] to perform up-and-down oscillating circular trajectories. Based on the intrinsic attitude stability of the hexarotor, we can consider that there is no rotational component measured by the optic flow sensors. Furthermore, we consider that pitch and roll are negligible despite the circular bouncing trajectory. Thus, the downward translational optic flow can be measured along the x component of the optic flow sensors. Position and orientation used in the hexarotor's control were taken from the motion-capture (MoCap) system installed in the Mediterranean Flying Arena. The flying arena was equipped with 17 motion-capture cameras covering a 6 × 8 × 6 m volume using a VICON™ system. Datasets including the optic flow measurements were recorded via the Robot Operating System (ROS) and processed with the Matlab/Simulink 2022 software.

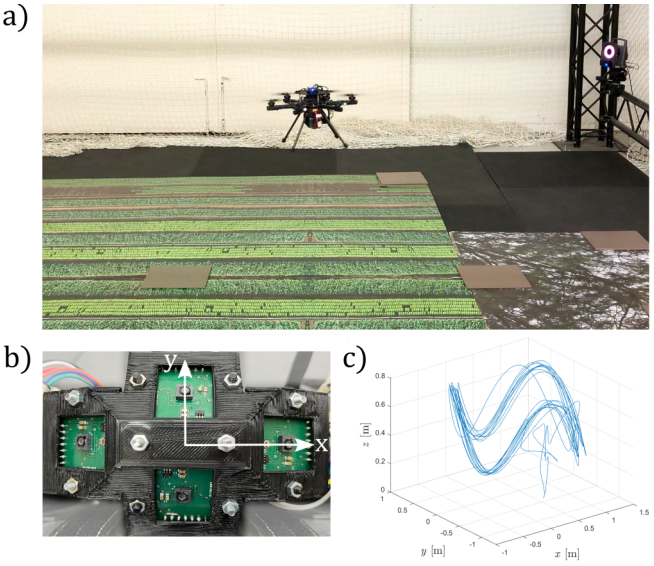


Figure 2: a) Hexarotor equipped with 4 optic flow sensors oriented towards the ground flying along a bouncing circular trajectory in the Marseille's flying arena. b) 2 optic flow sensors were set along the longitudinal axis x at angles $\phi = \pm 30°$ with respect to the hexarotor's vertical axis z, while the other 2 optic flow sensors were set along the lateral axis y at angles $\phi = \pm 30°$ with respect to the hexarotor's vertical axis z. c) Example of a test flight trajectory over $53m$ at an oscillation frequency of $0.28Hz$.

State space representation used for the EKF To estimate the hexarotor's flight height $\hat{h}$, we chose to model the hexarotor's system as a double integrator receiving as input the acceleration a_z on the vertical axis z given by the drone's IMU. Thus, the hexarotor's state space representation can be ex-

[1] https://github.com/gipsa-lab-uav/trajectory_control

http://www.imavs.org/

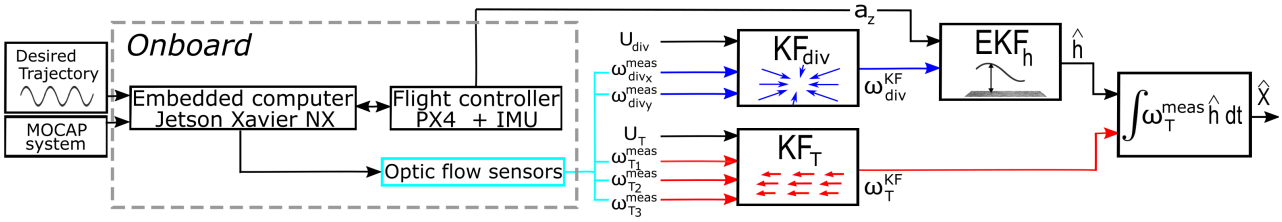


Figure 3: The sensor fusion based on 4 optic flow sensors rely on additional Kalman Filters (KF). The embedded computer handles the 4 optic flow sensors set on the hexarotor. $\omega_{div_x}^{meas}$ and $\omega_{div_y}^{meas}$ are taken as measurements by a KF (noted KF_{div}), that takes as input the current value U_{div} of the model of the optic flow divergence. The output of the KF is the local optic flow divergence ω_{div}^{KF}. $\omega_{T_1}^{meas}$, $\omega_{T_2}^{meas}$ and $\omega_{T_3}^{meas}$ are taken as measurements by a KF (noted KF_T), that takes as input the current value U_T of the model of the translational optic flow. The output of the KF is the local translational optic flow ω_T^{KF}. The EKF takes as input the hexarotor's acceleration a_z and as measurement ω_{div}^{KF} to estimate the current flight height $\hat{h}$. The EKF output $\hat{h}$ scales ω_T^{KF}, that is then integrated to perform odometry.

Specifics	Optic flow sensors
Sensor chip	Pixart PAW3903
Sensor PCB	4 × 2g
Hardware read-out of the 4 sensors	Arduino Nano

Table 1: Table of the specifics of the optic flow sensors equipped on the hexarotor.

pressed as:

$$\begin{cases} \dot{X} = f(X, a_z) = A \cdot X + B \cdot a_z = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \cdot X + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot a_z \\ Y = g(X) = [X(2)/X(1)] = V_h/h = \omega_{div} \end{cases} \tag{6}$$

where $X = \begin{bmatrix} h, V_h \end{bmatrix}^T$ is the hexarotor's state vector.
The use of an EKF was necessary due to the non-linearity of the local optic flow divergence, as the measurement depends on the ratio of both states V_h and h (see equation (3)).

5 ODOMETRY METHOD WITH 2 OPTIC FLOW SENSORS WITH NO PRIOR KNOWLEDGE (NPK)

The local optic flow divergence ω_{div}^{2S} was measured as the subtraction between the two raw optic flow magnitudes perceived by the two optic flow sensors set along the x axis, while the local translational optic flow ω_T^{2S} was measured as their sum. To estimate the flight height $\hat{h}$, the EKF received as:

- input: the acceleration of the drone a_z,
- measurement: the local optic flow divergence ω_{div}^{2S}.

See Appendix B for the EKF calculations.
$\hat{h}$ was then used to scale the integration of the local translational optic flow ω_T^{2S} to perform odometry. This odometry method based on 2 raw optic flow measurements does not need prior knowledge of any parameter to assess the distance traveled.

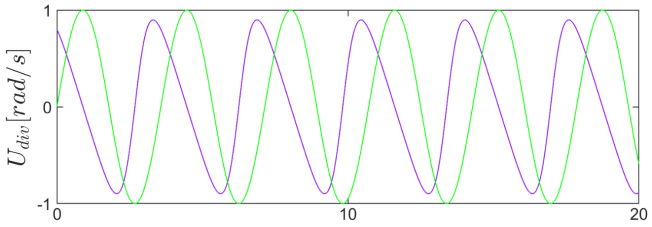


Figure 4: Inputs U_{div} of the Kalman Filters (KF) used to fuse optic flow divergence cues with the **P**recise **P**rior **K**nowledge (PPK) fusion strategy (in purple) and with the **R**ough **P**rior **K**nowledge (RPK) fusion strategy (in green), respectively. In the RPK sensor fusion strategy, only a sinus helps the KF to keep the oscillation timing as the shape looks similar.

6 FUSION STRATEGIES WITH 4 OPTIC FLOW SENSORS

6.1 *Fusion strategy using a very **P**recise **P**rior **K**nowledge (PPK) of the optic flow variations*

Here, we investigated how to use prior knowledge of the self-oscillation to further improve the accuracy of the distance traveled estimates.
We expressed the optic flow divergence induced by the self-oscillation to serve as input of a Kalman Filter (KF) as follows (see Figure 4):

$$\omega_{div} = \frac{\dot{h}}{h} \longrightarrow U_{div}(k) = \frac{A_{osc} 2\pi f_{osc} \cos(2\pi f_{osc} k\delta t)}{h_0 + A_{osc} \sin(2\pi f_{osc} k\delta t)} \tag{7}$$

with f_{osc} oscillation frequency equal to $0.28Hz$, A_{osc} oscillation amplitude equal to $0.25m$ and h_0 average flight height equal to $0.55m$. To fuse $\omega_{div_x}^{meas}$ and $\omega_{div_y}^{meas}$, we used a KF (see Figure 3). At each k^{th} step, the KF received as input the current value of the model in equation (7) and as measurements $\omega_{div_x}^{meas}$ and $\omega_{div_y}^{meas}$. See Appendix A for the KF calculations.

We expressed the translational optic flow induced by the

http://www.imavs.org/

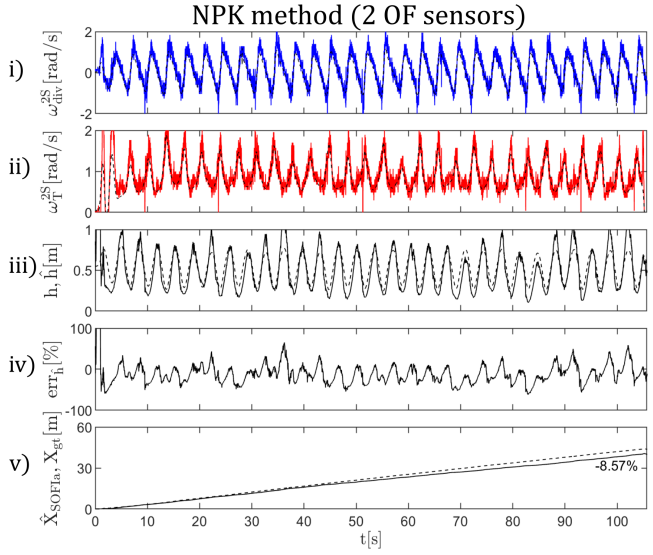


Figure 5: i) The local Optic Flow (OF) divergence ω_{div}^{2S} (in blue) measured with the **N**o **P**rior **K**nowledge (NPK) method had a Signal-to-noise Ratio (SnR) of $5.62dB$. The theoretical local optic flow divergence was computed as the ratio between V_h and h (in dashed line). ii) The local translational optic flow ω_T^{2S} (in red) measured with the NPK method had a SnR of $19.12dB$. The theoretical local translational optic flow was computed as the ratio between V_x and h (in dashed line). iii) The estimates of the flight height $\hat{h}$ converged quickly (within $4s$) to the ground truth h. iv) The average percentage error of $\hat{h}$ with respect to h after convergence was -9.77% (with a range of $[-61.5\%, 65.34\%]$). v) The final percentage error in the estimates of the distance traveled $\hat{X}_{SOFIa}$ with respect to the ground truth X_{gt} was -8.57%.

forward motion to serve as input of a KF as follows:

$$\omega_T = \frac{V_x}{h} \longrightarrow U_T(k) = \frac{\omega_T^{KF}(k-1) \cdot \hat{h}(k-1)}{h_0 + A_{osc}\sin(2\pi f_{osc} k\delta t)} \quad (8)$$

$V_x(0) \approx \omega_T^{KF}(k=0) \cdot \hat{h}(k=0)$ was initialized at $0.45m/s$. To fuse the three translational optic flow cues $\omega_{T_1}^{meas}$, $\omega_{T_2}^{meas}$ and $\omega_{T_3}^{meas}$, we used a KF (see Figure 3). At each k^{th} step, the KF received as input the current value of the model in equation (8) and as measurements $\omega_{T_1}^{meas}$, $\omega_{T_2}^{meas}$ and $\omega_{T_3}^{meas}$. See Appendix A for the KF calculations.

6.2 *Fusion strategy using a very **R**ough **P**rior **K**nowledge (RPK) of the optic flow variations*

Here, we investigated how to implement the sensor fusion strategy with 4 optic flow sensors without the knowledge of the oscillation amplitude A_{osc} and the average flight height h_0 just by using the knowledge of the trajectory oscillation timing.
To do so, we approximated very roughly both the optic flow divergence and the translational optic flow cues to a sinusoidal signal to serve as input of both KFs as follows (see Figure 4):

$$U_{div}(k) = U_T(k) = \sin(2\pi f_{osc} k\delta t) \quad (9)$$

with f_{osc} oscillation frequency equal to $0.28Hz$. At each k^{th} step, both KFs received as input the current value of the model in equation (9) and as measurements the divergence optic flow measurements ($\omega_{div_x}^{meas}$ and $\omega_{div_y}^{meas}$) and the translational optic flow measurements ($\omega_{T_1}^{meas}$, $\omega_{T_2}^{meas}$ and $\omega_{T_3}^{meas}$), respectively. See Appendix A for the KF calculations.

6.3 *Extended Kalman Filter within the fusion strategy with four optic flow sensors*

To estimate the drone's flight height $\hat{h}$, we used an EKF that received as:

- input: the acceleration of the drone a_z,
- measurement: the local optic flow divergence ω_{div}^{KF} filtered by the KF based on the measurements of the optic flow sensors.

See Appendix B for the EKF calculations.
$\hat{h}$ was used to scale the local translational optic flow ω_T^{KF}, that was then integrated to perform odometry as follows:

$$\hat{X}_{SOFIa} = \int \omega_T^{KF} \cdot \hat{h}\, dt \quad (10)$$

7 RESULTS

We compared the sensor fusion strategies based on **P**recise **P**rior **K**nowledge (PPK) and on **R**ough **P**rior **K**nowledge (RPK) of optic flow variations (using here 4 optic flow sensors) to the method based on **N**o **P**rior **K**nowledge (NPK) of optic flow variations (using here 2 optic flow sensors). 7 bouncing circular test flights of about $50m$ were performed with the hexarotor both under an illuminance of $117lux$ ($5.36 \cdot 10^{-6} W/cm^2$) and under an illuminance of $1518lux$ ($2.71 \cdot 10^{-4} W/cm^2$), for a total of 14 test flights. First, the 14 datasets were processed with the NPK method (see Section (5)). Then, the 14 datasets were processed using the PPK strategy (see Section (6.1)) and the RPK strategy (see Section (6.2)). The KF parameters defined in Appendix A were chosen as $\Phi = 10$, $\Gamma = 10$ and $H_k = 10$. The KF parameters were experimentally chosen with the first dataset taken under an illuminance of $1518lux$ and were used to process all 14 datasets for both PPK and RPK fusion strategies.
In Figures 5 and 6, the optic flow measurements were processed with the three strategies (NPK, RPK and PPK) for the same dataset taken under an illuminance of $1518lux$. The increase in the Signal-to-noise Ratio (SnR, computed as the squared ratio of the root mean square of the signal and the root mean square of its noise) observed in Figure 6 for the local optic flow divergence and the local translational optic flow between the NPK method and the PPK and RPK strategies influenced the average percentage error of the estimates

http://www.imavs.org/

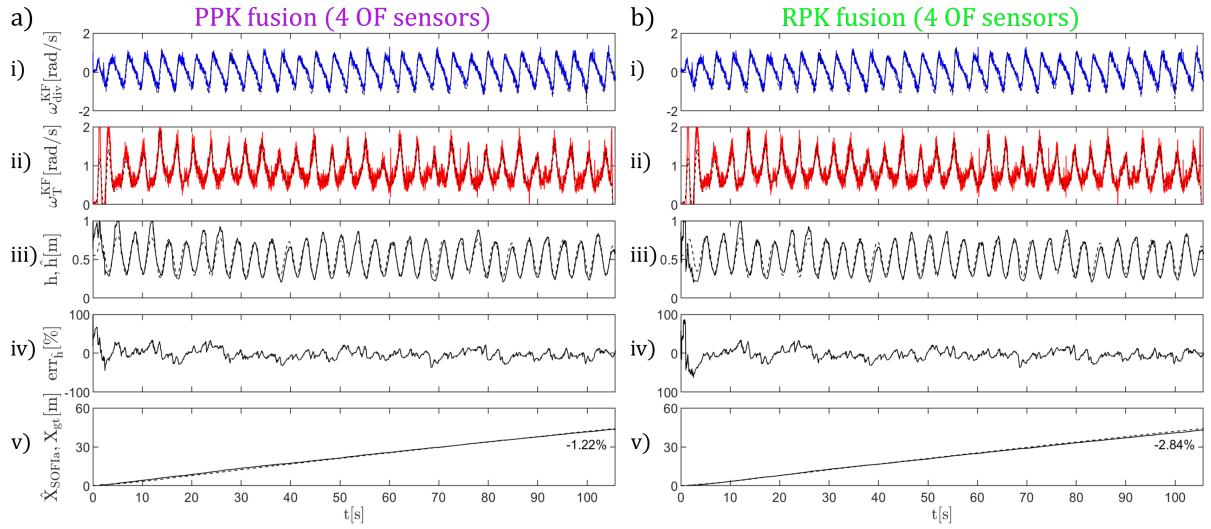


Figure 6: The local Optic Flow (OF) divergence ω_{div}^{KF} (in blue) measured with the **P**recise **P**rior **K**nowledge (PPK) strategy had a Signal-to-noise Ratio (SnR) of $6.72dB$ (a.i) and with the **R**ough **P**rior **K**nowledge (RPK) strategy of $6.73dB$ (b.i). The theoretical local optic flow divergence was computed as the ratio between V_h and h (in dashed line). The local translational optic flow ω_T^{KF} (in red) measured with the PPK strategy had a SnR of $25.74dB$ (a.ii) and with the RPK strategy of $25.91dB$ (b.ii). The theoretical local translational optic flow was computed as the ratio between V_x and h (in dashed line). The estimates of the flight height $\hat{h}$ converged quickly (within $4s$) to the ground truth h in both cases (a and b.iii). The average percentage error of $\hat{h}$ with respect to h after convergence was -2.16% for the PPK strategy (with a range of $[-36.89\%, 34.13\%]$) (a.iv) and -2.58% for the RPK strategy (with a range of $[-36.89\%, 34.23\%]$) (b.iv). The final percentage error in the estimates of the distance traveled $\hat{X}_{SOFIa}$ with respect to the ground truth X_{gt} was -1.22% for the PPK strategy (a.v) and -2.84% for the RPK strategy (b.v).

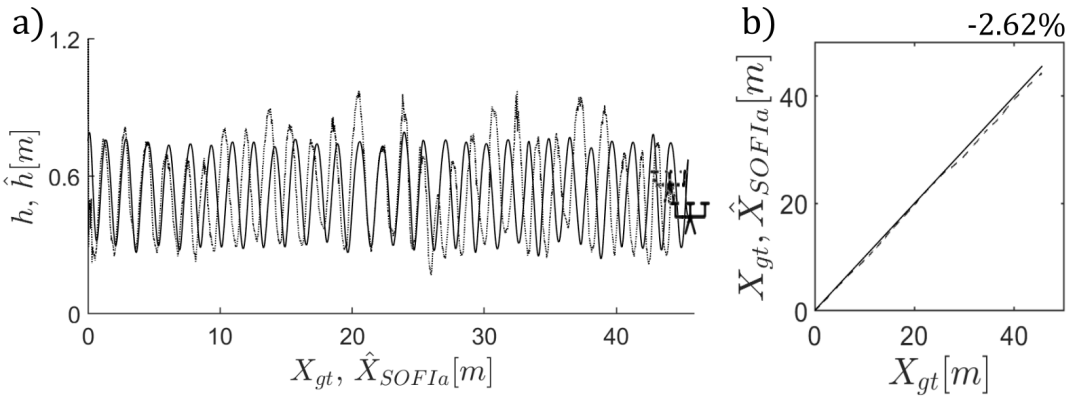


Figure 7: a) Comparison of the position of the hexarotor in the vertical plane (x,z) estimated with the **R**ough **P**rior **K**nowledge (RPK) fusion strategy (dashed line) to the ground truth given by the MoCap system (continuous line). The estimates of the flight height $\hat{h}$ were plotted on the estimates of the traveled distance $\hat{X}_{SOFIa}$, while the ground truth h was plotted on X_{gt}. This test flight was performed at $1518lux$. b) The final percentage error in the estimates of the traveled distance $\hat{X}_{SOFIa}$ with respect to the ground truth X_{gt} was -2.62%.

of the flight height after convergence (considered at $4s$). The flight height error was -9.77% for the NPK method (see figure 5), while it was -2.16% for the PPK strategy and -2.84% for the RPK strategy. Similar results were obtained for all 14 datasets. The SnR of the local translational optic flow measured with the NPK method ranged between $18.08dB$ and $24.79dB$, while with the PPK strategy it ranged between $24.84dB$ and $29.93dB$ and with the RPK strategy it ranged between $24.84dB$ and $28.64dB$. The SnR of the local optic flow divergence measured with the NPK method ranged between $5.41dB$ and $5.71dB$, while with both the PPK and RPK strategies it ranged between $6.47dB$ and $7.1dB$.

http://www.imavs.org/

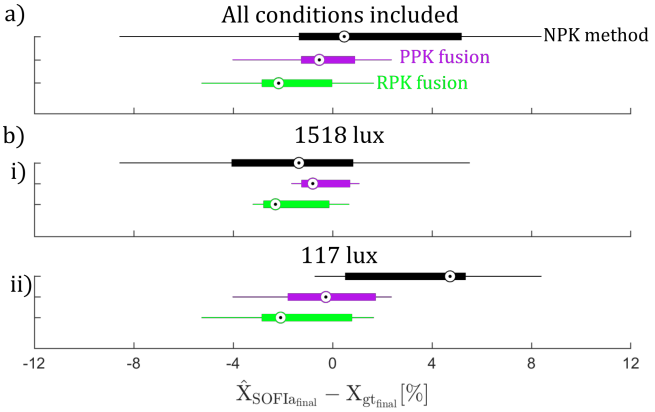


Figure 8: a) When considering all 14 datasets taken, the final percentage error ranged between -8.57% and 8.4% (with a median of 0.47%) for the **N**o **P**rior **K**nowledge (NPK) method (in black), between -4.02% and 2.38% (with a median of -0.53%) for the **P**recise **P**rior **K**nowledge (PPK) strategy (in purple) and between -5.27% and 1.66% (with a median of -2.17%) for **R**ough **P**rior **K**nowledge (RPK) (in green). b.i) At $1518lux$, the final percentage error ranged between -8.57% and 5.52% (with a median of -1.14%) for the NPK method, between -1.65% and 1.08% (with a median of -0.8%) for the PPK strategy and between -3.21% and 0.67% (with a median of -2.3%) for RPK strategy. b.ii) At $117lux$, the final percentage error ranged between -0.72% and 8.4% (with a median of 4.73%) for the NPK method, between -4.02% and 2.38% (with a median of -0.27%) for the PPK strategy and between $-5.27\%\%$ and 1.66% (with a median of -2.09%) for the RPK strategy.

The computation of the estimates of the flight height $\hat{h}$ and of the estimates of the traveled distance $\hat{X}_{SOFIa}$ allow to assess the position of the hexarotor in the vertical plane (x,z). An example is shown in Figure 7, in which the estimates of the flight height $\hat{h}$ were plotted on the estimates of the traveled distance $\hat{X}_{SOFIa}$ (which are given directly in meters) and compared to the ground truth given by the MoCap system. Since it is based on the optic flow based odometry, the 2D position estimation is subject to an accumulation of error increasing with the distance covered.

Overall, the final percentage error in the estimates of the distance traveled $\hat{X}_{SOFIa}$ with respect to the ground truth X_{gt} (traveled along the x axis) ranged between -8.57% and 8.4% for the NPK method, between -4.02% and 2.38% for the PPK strategy and between -5.27% and 1.66% for the RPK strategy (see Figure 8.a). Similar results were obtained when considering the two different illuminances separately (see Figure 8.b).

8 CONCLUSION

In this study, we investigated how to use some knowledge of the oscillating trajectory to improve a minimalistic odometry based on optic flow cues. The experiments were performed onboard a hexarotor following circular bouncing trajectories at a frequency of $0.28Hz$ over distances of about $50m$ under illuminances of $117lux$ and $1518lux$. Results were not influenced by illuminance conditions.

Our findings show that the sensor fusion strategies based on the use of 4 optic flow sensors allowed to measure the optic flow divergence and the translational flow cues more reliably thanks to filtering made by additional Kalman Filters. This was the case even when taking into consideration only a rough prior knowledge of the optic flow variations and more specifically only the oscillation timing of the trajectory. This prior knowledge can be considered acceptable since the oscillation timing is imposed by the drone itself on its own forward trajectory. The sensor fusion strategies decreased the error in the estimates of the flight height. Consequently, they decreased the percentage error in the estimates of the distance traveled in every case considered and thus improved odometry performance.

For all three methods, we acknowledge that the final traveled distance estimates are subject to a small error as the odometry strategy is a dead reckoning method without any feedback from the environment. Such a minimalistic optic flow based odometry strategy would allow a future drone to assess whether it is returning near its base station without GPS. So far, our findings can be considered as a first experimental proof-of-concept of the SOFIa model [23] before implementing such optic flow based odometry strategy on a nanodrone relying on low-computational power, as considered in [26]. Furthermore, we need to validate the robustness of these strategies in a range of forward speeds, in case of large drone pitch, in the presence of reliefs and finally outdoors.

Future work will also include the implementation of an optic flow regulator to keep the translational optic flow around a given setpoint.

ACKNOWLEDGMENT

We thank J.M. Ingargiola for his help in the design of the printed circuit boards for the optic flow sensors. Financial support was provided via a ProxiLearn project grant to F.R. from the ANR (Astrid Program). The participation of L.B. in this research project was supported by a joint PhD grant from the Délégation Générale de l'Armement (DGA) and Aix Marseille University. L.B, and F.R. were also supported by Aix Marseille University and the CNRS (Life Science, Information Science Institute as well as Engineering Science & Technology Institute). We are grateful to the three anonymous referees, whose suggestions helped us to greatly improved the manuscript.

REFERENCES

[1] R. Strydom, S. Thurrowgood, and M. V. Srinivasan. Visual odometry: autonomous uav navigation using op-

http://www.imavs.org/

tic flow and stereo. In Proceedings of Australasian conference on robotics and automation, 2014.

[2] V. More, H. Kumar, S. Kaingade, P. Gaidhani, and N. Gupta. Visual odometry using optic flow for unmanned aerial vehicles. In 2015 International Conference on Cognitive Computing and Information Processing(CCIP), pages 1–6, 2015.

[3] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza. Ultimate slam? combining events, images, and imu for robust visual slam in hdr and high-speed scenarios. IEEE Robotics and Automation Letters, 3(2):994–1001, 2018.

[4] A. Denuelle and M. V. Srinivasan. A sparse snapshot-based navigation strategy for uas guidance in natural environments. In 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 3455–3462. IEEE, 2016.

[5] F. Kendoul, I. Fantoni, and K. Nonami. Optic flow-based vision system for autonomous 3d localization and control of small aerial vehicles. Robotics and autonomous systems, 57(6-7):591–602, 2009.

[6] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza. Autonomous, vision-based flight and live dense 3d mapping with a quadrotor micro aerial vehicle. Journal of Field Robotics, 33(4):431–450, 2016.

[7] S. H. Lee and G. de Croon. Stability-based scale estimation for monocular slam. IEEE Robotics and Automation Letters, 3(2):780–787, 2018.

[8] R. Milijas, L. Markovic, A. Ivanovic, F. Petric, and S. Bogdan. A comparison of lidar-based slam systems for control of unmanned aerial vehicles. In 2021 International Conference on Unmanned Aircraft Systems (ICUAS), pages 1148–1154. IEEE, 2021.

[9] A. Shurin and I. Klein. Qdr: A quadrotor dead reckoning framework. IEEE Access, 8:204433–204440, 2020.

[10] F. Ruffier and N. Franceschini. Optic flow regulation: the key to aircraft automatic guidance. Robotics and Autonomous Systems, 50(4):177–194, 2005.

[11] F. Expert and F. Ruffier. Flying over uneven moving terrain based on optic-flow cues without any need for reference frames or accelerometers. Bioinspiration and Biomimetics, 10, 2015.

[12] F. Iida. Biologically inspired visual odometer for navigation of a flying robot. Robotics and autonomous systems, 44(3-4):201–208, 2003.

[13] F. Kendoul, I. Fantoni, and K. Nonami. Optic flow-based vision system for autonomous 3d localization and control of small aerial vehicles. Robotics and autonomous systems, 57(6-7):591–602, 2009.

[14] J. R Serres and F. Ruffier. Optic flow-based collision-free strategies: From insects to robots. Arthropod structure & development, 46(5):703–717, 2017.

[15] WH Kirchner and MV Srinivasan. Freely flying honeybees use image motion to estimate object distance. Naturwissenschaften, 76(6):281–282, 1989.

[16] G. Portelli, F. Ruffier, F. L. Roubieu, and N. Franceschini. Honeybees' speed depends on dorsal as well as lateral, ventral and frontal optic flows. PloS one, 6(5):e19486, 2011.

[17] B. Herissé, T. Hamel, R. Mahony, and F.-X. Russotto. Landing a vtol unmanned aerial vehicle on a moving platform using optical flow. IEEE Transactions on robotics, 28(1):77–89, 2011.

[18] F. Van Breugel, K. Morgansen, and M. H Dickinson. Monocular distance estimation from optic flow during active landing maneuvers. Bioinspiration & biomimetics, 9(2):025002, 2014.

[19] G. CHE de Croon. Monocular distance estimation with optical flow maneuvers and efference copies: a stability-based strategy. Bioinspiration & biomimetics, 11(1):016004, 2016.

[20] H. W. Ho, G. CHE de Croon, and Q. Chu. Distance and velocity estimation using optical flow from a monocular camera. International Journal of Micro Air Vehicles, 9(3):198–208, 2017.

[21] V. Creuze. Monocular odometry for underwater vehicles with online estimation of the scale factor. In IFAC 2017 World Congress, 2017.

[22] L. Bergantin, T. Raharijaona, and F. Ruffier. Estimation of the distance from a surface based on local optic flow divergence. In 2021 International Conference on Unmanned Aircraft Systems (ICUAS), pages 1291–1298. IEEE, 2021.

[23] L. Bergantin, N. Harbaoui, T. Raharijaona, and F. Ruffier. Oscillations make a self-scaled model for honeybees' visual odometer reliable regardless of flight trajectory. Journal of the Royal Society Interface, 18(182):20210567, 2021.

[24] J. J. Gibson. The perception of the visual world. 1950.

[25] L. Meier, D. Honegger, and M. Pollefeys. Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In 2015 IEEE

http://www.imavs.org/

International Conference on Robotics and Automation (ICRA), pages 6235–6240, 2015.

[26] D. Palossi, A. Marongiu, and L. Benini. Ultra low-power visual odometry for nano-scale unmanned aerial vehicles. In Design, Automation Test in Europe Conference Exhibition (DATE), 2017, pages 1647–1650, 2017.

APPENDIX A: KALMAN FILTER CALCULATIONS

For the PPK strategy, the optic flow divergence and translational optic flow cues were expressed as in equations (7) and (8), respectively. For the RPK strategy, the optic flow divergence and translational optic flow cues were both expressed as in equation (9). For each optic flow cue, at each k^{th} step the current value of the corresponding model was computed and given to the corresponding KF as input (see Figure 3). In the following paragraph, the notation $A > 0$ indicates a matrix strictly positive definite. The KF took the following iterative steps for each k^{th} time:

Prediction step

(a) One-step ahead prediction

$$X_{k/k-1} = \Phi \cdot X_{k-1/k-1} + \Gamma \cdot U_{k-1/k-1} \tag{11}$$

with $\Phi > 0, \Gamma > 0$.

(b) Covariance matrix of the state prediction error vector

$$P_{k/k-1} = \Phi \cdot P_{k-1/k-1} \cdot \Phi^T + Q \tag{12}$$

Correction step

(c) Measurement update

$$X_{k/k} = X_{k/k-1} + K_k \cdot (Y_k^i - H_k \cdot X_{k/k-1}) \tag{13}$$

with Y_k^i current value of the i^{th} measurement, $H_k > 0$ and K_k Kalman gain defined as:

$$K_k = P_{k/k-1} \cdot H_k^T \cdot [H_k \cdot P_{k/k-1} \cdot H_k^T + R_k]^{-1} \tag{14}$$

The measurement update step was repeated for each i^{th} measurement (2 times for the optic flow divergence and 3 times for the translational optic flow).

(d) Covariance matrix of state estimation error vector

$$P_{k/k} = P_{k/k-1} + K_k \cdot [H_k \cdot P_{k/k-1} \cdot H_k^T + R_k] \cdot K_k^T \tag{15}$$

(e) Innovation

$$\tilde{Y_k} = Y_k - H_k \cdot X_{k/k} \tag{16}$$

APPENDIX B: EXTENDED KALMAN FILTER CALCULATIONS

The discretized model of the hexarotor (equation (6)) can be expressed as

$$\begin{cases} X[k+1] = \Phi \cdot X[k] + \Gamma \cdot U[k] \\ Y[k] = C_k \cdot X[k] + D_k \cdot U[k] \end{cases} \tag{17}$$

with

$$\Phi = e^{A \cdot dt} \tag{18}$$

$$\Gamma = (\int_0^{dt} e^{A \cdot \tau} d\tau) \cdot B = (A^T \cdot e^{A \cdot dt} - A^T) \cdot B \tag{19}$$

$$C_k = h(X_k) = \begin{bmatrix} \frac{X_2[k]}{X_1[k]} \end{bmatrix} \tag{20}$$

$$D_k = 0 \tag{21}$$

where dt is the discretization time. To estimate the flight height h, the EKF took the following iterative steps for each k^{th} time:

Prediction step

(a) One-step ahead prediction

$$X_{k/k-1} = \Phi \cdot X_{k-1/k-1} + \Gamma \cdot U_{k-1/k-1} \tag{22}$$

(b) Covariance matrix of the state prediction error vector

$$P_{k/k-1} = \Phi \cdot P_{k-1/k-1} \cdot \Phi^T + Q \tag{23}$$

Correction step

(c) Measurement update

$$X_{k/k} = X_{k/k-1} + K_k \cdot (Y_k - H_k \cdot X_{k/k-1}) \tag{24}$$

with K_k Kalman gain defined as:

$$K_k = P_{k/k-1} \cdot H_k^T \cdot [H_k \cdot P_{k/k-1} \cdot H_k^T + R_k]^{-1} \tag{25}$$

and H_k Jacobian matrix for the non linear function defined as follows:

$$H_k = \frac{\partial h}{\partial X}|_{X=X_{k/k-1}} = \begin{bmatrix} -\frac{\dot{h}}{h^2} & \frac{1}{h} \end{bmatrix} \tag{26}$$

(d) Covariance matrix of state estimation error vector

$$P_{k/k} = P_{k/k-1} + K_k \cdot [H_k \cdot P_{k/k-1} \cdot H_k^T + R_k] \cdot K_k^T \tag{27}$$

(e) Innovation

$$\tilde{Y_k} = Y_k - H_k \cdot X_{k/k} \tag{28}$$

http://www.imavs.org/

Seeing with sound; surface detection and avoidance by sensing self-generated noise

Simon Wilshin, Stephen Amos, Richard J Bomphrey
Structure and Motion Lab, Royal Veterinary College
swilshin@rvc.ac.uk

http://www.imavs.org/

ABSTRACT

Here, we demonstrate obstacle and secondary drone avoidance capability by quadcopter drones that can perceive and react to modulation of their self-generated acoustic environment when in proximity to surfaces. A ground truth for the interpretation of self-noise was established by measuring the intrinsic, three-dimensional, acoustic signature of a drone in an anechoic chamber. This was used to design sensor arrangements and machine learning algorithms to estimate the position of external features, obstacles or another drone, within the environment. Our machine learning approach took short segments of recorded sound and their Fourier transforms, fed these into a convolutional neural network, and output the location of an obstacle or secondary drone in the environment. The convolutional layers were constructed with a suitable topology that matched the physical arrangement of the sensors. Our surface detection and avoidance algorithms were refined during tethered flight within an anechoic chamber, followed by an exercise in free flight without obstacle avoidance, and finally free flight obstacle detection and avoidance. Our acoustic sense-and-avoid capability extends to vertical and horizontal planar surfaces and tethered secondary drones.

1 INTRODUCTION

Many methods exist by which a drone might sense its environment, such as LIDAR and laser range-finders[1], cameras and optical flow sensors[2], pressure sensors [3], or even bioinspired "hair sensors"[4, 5]. Each such sensor modality has strengths and weaknesses.

Another method of environmental detection is acoustic sensing with microphones. These have been used to estimate the distance[6] and bearing[7] between drones and the relative location of drones[8] in a swarm. Drone mounted acoustic sensors, when integrated MEMS and particle velocity sensors, have been shown to be capable of detecting the location of large civil aircraft[9]. Ground based acoustic sensors have also been used to localise small gasoline-powered UAV over hundreds of meters[10].

Acoustic surface detection methods have enjoyed success on robotic platforms drawing inspiration from biological systems[11]. Echolocation has been used to relocate a robot in an environment by comparison with existing maps of said environment[12]. Active echolocation on has also been employed on the crazyflie drone to identify surfaces[13], though this requires a loud on-board speaker to generate the required sound, which we wish to avoid.

Flying drones generate sound, principally from the motors and the aeroacoustic noise from the propellors (which we term the acoustic signature), and nearby surfaces in the environment reflect that sound. The spatio-temporal structure of the reflected acoustic signature contains information about the position of the drone relative to obstacles in the environment. Microphones that observe this information can also detect the acoustic signature of other drones in the environment. While physics based acoustic modelling approaches have been shown to be theoretically viable for surface detection in real time[14], here we employ a machine learning approach.

This method of obstacle and drone detection offers several advantages over others. Unlike cameras, microphones work in the dark and draw minimal power; our arrays draw only a few mA. This makes them suitable even for microdrones weighing only a few tens of grams.

They can be used passively to listen for the sound of other drones, or actively by listening to the reflected sound of the drone's own acoustic signature. In either case, they make such a drone no more conspicuous than it would be operating in free flight, since a flying drone must operate its engines to resist gravity regardless of whether or not it is sensing the environment. This in contrast to laser range-finding or LIDAR; if a laser beam is directed at a suitable sensor it can be readily detected and the direction of the drone quickly determined.

The approach is also more general than sensing environmental changes by monitoring motor power consumption directly. Such sensing can be used to detect a surface below a drone and shut off the motors for landing (for example in[15]), but has limited utility for sensing vertical planar surfaces in the environment and no capability to detect other drones.

Here, we show how to use such information to detect surfaces, both in ideal conditions while tethered in an anechoic chamber, and in the variable, suboptimal conditions that occur in free flight. We then demonstrate how such information

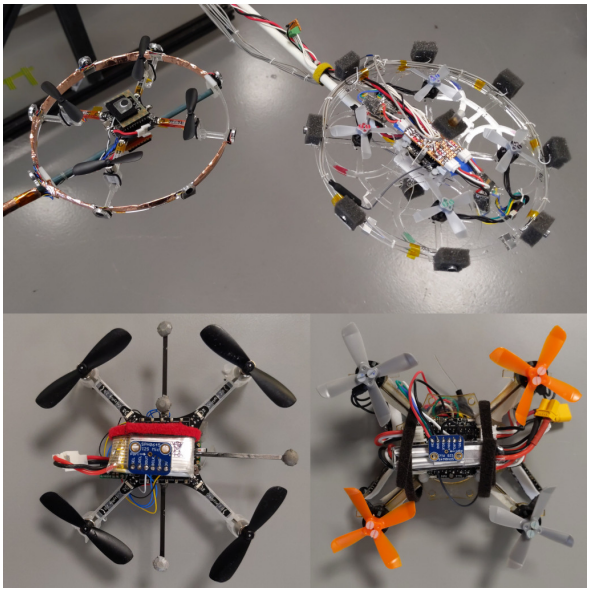


Figure 1: Various microphone arrays deployed on the Crazyflie and Mormoops prototype. At the top are the analogue microphone arrays deployed in a ring around the Crazyflie on the left and the Mormoops on the right. Below are the digital microphone arrays (in ground detection configuration with microphones on top of and below the drones body), again with the Crazyflie on the left and the Mormoops on the right). The bottom left drone was the final prototype used in the sense and avoid experiments

can be used to avoid collision with obstacles in free flight.

Finally, we demonstrate how such a sensor system can be used to detect another drone's location in the environment, while tethered, with a high degree of accuracy.

2 METHODS

2.1 Acoustic sensors

We have developed several acoustic sensor systems as required, including tethered analogue microphone arrays, on-drone analogue microphone arrays digitised by an onboard microcontroller, and onboard digital microphones managed by an onboard microcontroller ((Fig. 1)). While specialised small, directional acoustic sensors do exist we opted for an array of microphones similar to those that have already seen successful employment on drones[16]. Directional microphones were rejected because it was felt having multiple recordings on an array would allow for better detection of interference effects.

The outputs from tethered analogue microphone arrays were digitised during the characterisation of the acoustic signature of the Crazyflie, and for tethered surface detection in the anechoic chamber, and for the detection of anther drone in the environment when both drones were tethered.

Following measurements of the acoustic signature of the drones and the tethered trials, for our recordings in free flight we used digital I_2S microphones connected to a teensy 4.0. This permitted the fast onboard notch filtering and power estimation of recordings needed for in flight surface detection and avoidance.

2.2 Self-generated drone acoustic signature

In order to design and calibrate our surface detection algorithms for our drone, we needed to to obtain the self-generated acoustic signature: that is, the sound the drone made in various directions while hovering.

To do this, we mounted an array of analogue acoustic microphones on a Crazyflie and recorded the sound generated at a fixed distance at a range of angles (elevations at $45\,\mathrm{deg}$ intervals and at 18 azimuthal angles approximately $15\,\mathrm{deg}$ seperation, $n = 90$) arranged around the drone. Note these arrays are not the same as those fixed to the drone, as can be seen in figure 2, these microphones are attached to the fixed white semicircle suspended in the chamber. Recordings were taken with 6mm diameter uni-directional electret condenser microphones (RS PRO, 50 to 16000 Hz at -47dB) wired to separate signal conditioning amplifiers (Vishay 2210B), and a Powerlab DAQ (PowerLab 16/35, ADInstruments).

We took recordings in an anechoic chamber (Fig. 2) to maximise the quality of the acoustic signature signals, shown in Fig. 5.

2.3 Tethered surface detection using self-generated sound

Based on the results from the self-generated acoustic signature of our drones(Fig. 5), we designed a microphone array for detecting surfaces in the environment with a high density of sensors at eight different azimuthal angles and two sensors above and below the drone (Fig. 1, top right). We used this array to detect detect planar surfaces while tethered in an anechoic chamber using self-generated noise.

Based in part on the results from the acoustic signature of the drone we decided to implement a machine learning classifier built using PyTorch[17]. The classification network is built from a configurable input layer, two convolutional layers, two fully connected layers and an output layer.

The input layer can be adjusted to match the sensor suite being used via a human readable YAML configuration file. In this configuration the data is split into blocks.

These blocks are the microphone recordings from the eight sensors in a ring, the two microphone on the top and bottom of the drone, and two additional blocks formed from the Fast Fourier Transform (FFT) of these two sets of signals. The FFT was taken along the time axis of each block using the rfft function of tensor flow.

Two convolutional layers are then used. A useful summary of recent applications is provided in [18] . The surrounding sensors form a ring around the drone, and any classification network should respect this topology, i.e. there should be no 'start' or 'end' sensor, and any mathematical

http://www.imavs.org/

http://www.imavs.org/

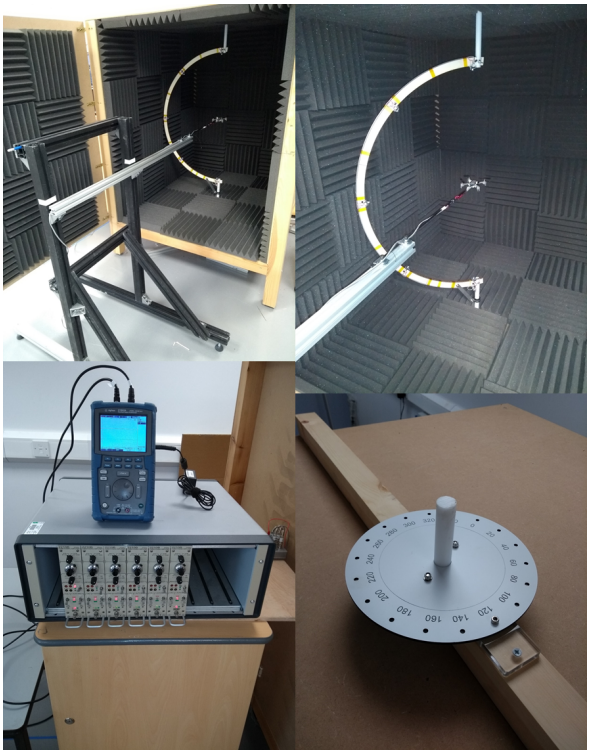


Figure 2: Apparatus for measuring quadcopter acoustic signature. The arc is fitted with five microphones and can be rotated in azimuth to repeatable angles around the tethered quadcopter. Microphones are mounted at the North and South poles, the equator and +/- 45 degrees. The microphones are fed by common excitation and the output signal is received by individual Vishay 2210B signal conditioning amplifiers. The quadcopter motors are provided with a voltage known to support the weight of the aircraft.

operation should be invariant to a transformation which shifts all the sensors around this ring.

As such we use a circular convolution for the sensor input elements on the ring. Since the top and bottom elements of the microphone are completely included in the convolution along the sensor number axis, these can be a regular convolution.

If x_ij is an element of the blocks, x, discussed above, with i running from one to the sample length N, that is if we take 20 millisecond blocks at 22 kilohertz this would be 440 samples, j running from one to the number of elements in this block (for example eight if we are dealing with the eight microphones around the drone), then this layer performs the following operation:

$$y_\mu = b + w_\mu \star x \tag{1}$$

Where $\star$ is the cross-correlation operator, w_μ is the kernel associated with the μth convolution, of which there are M total, and b is a constant matrix. In the language of convolutional neural networks, this first layer assumes inputs with only one channel, and outputs M channels.

All units in the network are Rectified Linear Units [19] (ReLU) to improve training ease training and speed convergence, and so this response y is modified by:

$$y'_\mu = \max(0, y_\mu) \tag{2}$$

Our network pools after every convolutional layer to generate invariant features, which makes the network less sensitive to the precise locations of features and reduces the number of parameters in the network [20], reducing the risk of overfitting. We found that two-dimensional max pooling worked well, splitting the layer activations into $K \times L$ blocks and transforms the layer by downsampling to only the maximum value in these blocks.

Our network then proceeds with another convolutional layer with ReLU and pooling. Next, we have two fully connected layers. We treated the output of the last convolutional layer as a single vector a_i. The activation b_j of the fully connected layer is

$$b_j = c_j + \sum_i m_{ji} a_i \tag{3}$$

The units in the fully connected layer are again ReLU units, for the same reason as previously discussed. No pooling is performed here as the fully connected layer respects the topology of the input only in the most trivial way (every unit is connected to every input), meaning neighbouring units are likely to be unrelated.

Another fully connected layer is then used followed by the output layer. The output layer is, for this case, the azimuthal angle of the obstacle, and a vector pointing to the obstacle in the plane. The azimuthal angle can only be 0 or 90 degrees, and is re-interpreted after the network is trained as a class variable indicating wall position.

http://www.imavs.org/

The network is trained using Stochastic Gradient Descent (SGD) [21] to minimise the mean square error.

The network performance was evaluated on a test data set of ten thousand sections two milliseconds in length that were not used in the training data. As the testing and training data are drawn randomly from sixty second length trials there is a risk that temporal correlation within the longer trials could allow for identification of the sample.

2.4 Surface detection in free flight

As a precusor to surface avoidance we created a version of our prototype drone with the ability to detect surfaces in free flight. The prototype (called Mormoops) consisted of a Bolt flight controller *(Bitcraze, Malmö, Sweden)*, a custom chasis and four Betafpv 1105 5000KV brushless motors *Betafpv, Hong Kong, PRC)*. The drone was operated near a planar surface, and our neural network tasked with detecting the surface.

A Centeye optical flow sensor, *(Centeye, Washington, DC, USA)* was used to measure the change in height of the drone over one minute trials. The flight controller maintained constant height based on optical flow odometry and no obvious drift was observed over these short flights. The drone was flown at three different heights, one deep in ground effect at approximately 5cm height and two outside at 15 and 20cm.

Surface detection in free flight benefits from higher bandwidth microphone recording and processing. As such we decided to use a digital microphone platform combining SPH0645LM4H I^2S microphones *(Adafruit, New York City, USA)* with a Teensy 4.0) microcontroller *(PJRC, Sherwood, OR, USA)* with custom SD card reader for storage.

This would allow us to use the Teensy microcontroller's specialised I^2S audio processing hardware that enables high quality recording, fast filtering, Fourier transforms, and other signal processing.

Our neural network was then required to distinguish these states using the sound recordings from $20ms$ samples.

2.5 Free flight surface avoidance

Tests of wall detection capabilities were performed during the previous free flight experiments. However, this version of our drone could not safely resist wall effect, so insufficient training data could be gathered.

We modified our algorithms to detect and avoid both horizontal (ground) and vertical (wall) planar surfaces in free flight. We used the difference in power of the recording from two notch filtered microphones, a method that was based on examination of the structure of the machine learning approach - specifically the frequency dependency - and required minimal tethered training data.

The controller requires platform-specific calibration. The frequency range of the notch filter is specific to the drone that carries the sensor package, as are the gains on the microphones, and the threshold for detection. Ground detection is insensitive to these parameters, with simple benchtop tests sufficient for a good calibration. Wall detection requires in-flight refinement and iteration and is less reliable. In both instances the threshold for engaging the avoidance behaviour had to be set prior to the test.



Figure 3: Image of the autonomous flight arena during its early development showing Qualisys motion tracking cameras and a Crazyflie UAV on the floor in the centre of a netted enclosure. This was used to track and control the Crazyflie during sense-and-avoid flight trials.

At present, for avoidance purposes the system behaves as a classifier (the conditions being "wall present" and "no wall"). For the ground plane this is a simplification of the multiple heights that can be discerned by the machine learning controller, but for wall detection little evidence of continuous signal was apparent beyond the simple detection of wall effect.

Once an estimate of the surface position is available on the Teensy microcontroller, the information is communicated to the onboard Crazyflie flight controller by a parallel digital on the GPIO pins (Fig. 1). This distance estimate is then transmitted, via the Crazyradio, to a computer that tracks the Crazyflie using a Qualisys motion capture system *(Qualisys, Göteborg, Sweden)*. (Fig. 3)

If an obstace is detected, evasive action is taken to reposition the drone by updating the target position set on the control computer.

2.6 Sensing other drones

We determined to what extent it was possible to identify a second drone using the same acoustic sensor array. Initially, we confined our considerations to detecting drones that are physically different, the Crazyflie and our new Mormoops drone.

We ran both drones simultaneously in tethered flight and recorded the acoustic signature of each at a range of distances and angles, as in Fig. 4. The objective was to use our microphone array and machine learning approach to detect and locate the secondary drone, from the observer drone.

The analogue array was used to record from ten microphones distributed as for tethered surface detection. Recordings were performed in the same manner, though outside of

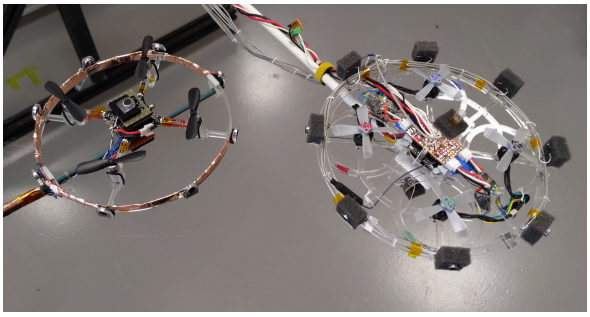


Figure 4: Experimental configuration for detecting and localising a prototype Mormoops drone from a Crazyflie, and *vice versa*. Recordings were taken from the microphone array on each drone, and used in conjunction with our machine learning approach to identify and localise each drone.

the anechoic chamber due to space constraints.

3 Results

3.1 Self-generated drone acoustic signature

Acoustic signatures are shown in Fig 5. Position dependent differences are apparent in these signals, which suggested that discerning positional information about quadcopters from external acoustic recordings was highly feasible. The spatial distribution of the signal suggests the need for multiple sensors around the body of the drone. These results suggested, and later observations confirmed, that a single pair of microphones mounted above and below the body would be sufficient in these directions for planar surface detection.

3.2 Tethered surface detection through self-generated acoustics

Performance of the network was excellent (see Fig. 6 and Fig. 7).

The drone was able to distinguish perfectly between the two conditions (wall to the side, floor below) for all examples in the test data set. When estimating the height of the drone above the surface the standard error was 36.1mm. This error was relatively consistent over the range of distances measured, meaning that when deep in ground effect the detection of the ground was relatively inaccurate (although amply sufficient to trigger a surface avoidance response), and surprisingly good at large distances (although we note that these constitute close to ideal conditions). The limits of this floor detection capability were not found as we reached the maximum height of the anechoic chamber before such a limit was reached. For practical purposes this is sufficient, although detecting such a limit for a free-flying drone is highly desirable and is a priority future objective.

Similarly, the horizontal distance at which performance begins to degrade was not detected, as the algorithm maintained a good performance out to 250mm, with a standard error on the distance to the wall at 39.5mm and an angular

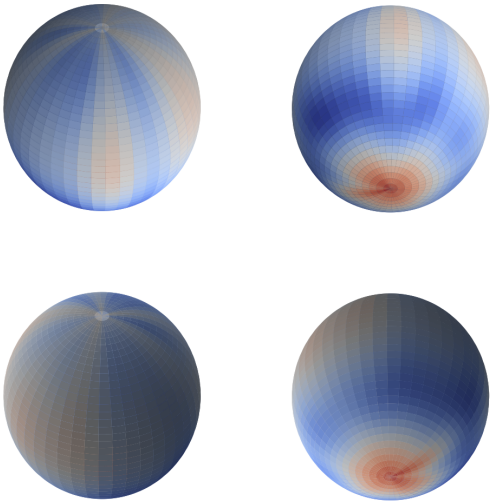


Figure 5: The external acoustic signature of a crazyflie. Here is shown a sphere around the drone and the acoustic power recorded at that location around the drone is displayed, cool colours indicate low acoustic power while hot colours indicate high acoustic power. The engines generate high pressure below the drone, and this can be seen as the red region on the bottom of the sphere. The four sphere are four different views, left and right are different elevations and the plots are spheres are rotated half a circle azimuthally going from the top to bottom row. The covariance matrix used in our anomaly detection algorithm can likely be inferred (or at least constrained) through these observations. This has the virtue that a physically derived estimate is guaranteed positive definite.

standard error (assuming a Gaussian distribution, rather than von Mises, which is a reasonable approximation here as the error is small), of 16.4 deg.

3.3 Surface detection in free flight

For the trials in ground effect, the sound recordings were clearly different, close approach to surfaces can be reliably detected from acoustic signatures. The drone was set to hover at three discrete heights, 5cm, 15cm and 20cm. The diameter of the rotos was 5.08mm. For the case where the drone was at 5cm height, deep in ground effect, 96% of trials were successfully identified. For the 15cm and 20cm cases 46% and 48% were successfully identified. Close detection of planar horizontal surfaces is reliable, especially if integrated over several seconds.

3.4 Free flight surface avoidance

The drone sucessfully takes action to avoid collisions with planar surfaces. It manoeuvres away from them as they are brought close. Examples of such behaviour can be seen in

http://www.imavs.org/

http://www.imavs.org/

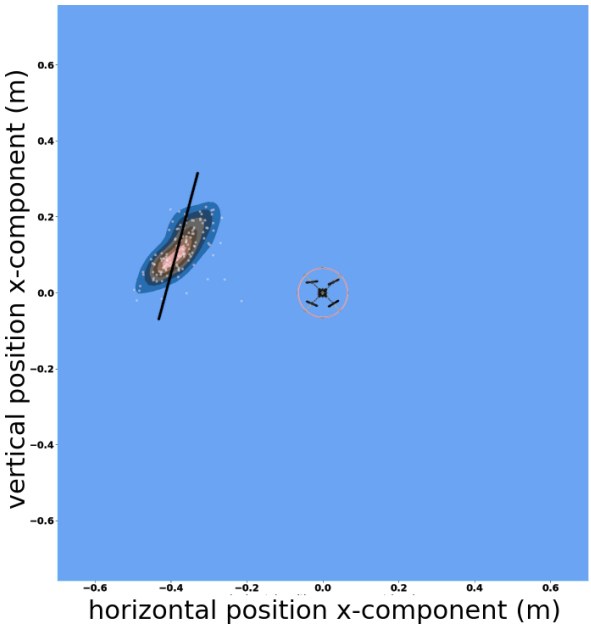


Figure 6: Performance of the machine learning sensory integrator in detecting the position of a surface to the side of the drone. The drone is shown inside the red circle, the wall is the solid black line, the x- and y-axis are position in meters. The white dots show the estimated position of the wall according to the algorithm, with the kernel density plot [22] in blue, black and red constructed from this data. There is strong agreement between these estimates and the actual wall position. The standard error in the estimated position by shortest distance to the wall was 39.5mm, and the standard error on the angle was 16.4 deg

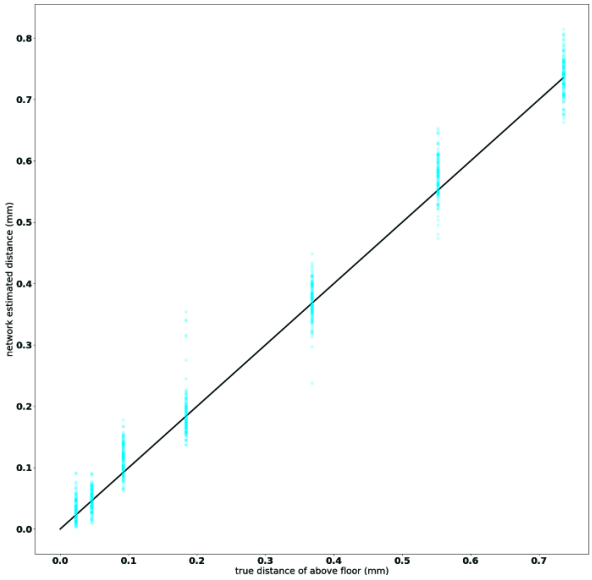


Figure 7: Performance of the machine learning sensory integrator in detecting the position of a surface to the below of the drone. On the x-axis is the true position of the surface below the drone in meters. On the y-axis is the estimated position of the surface according to the algorithm. The standard error on the estimated distance was 36.1mm.

the in Fig. 8 and supplemental movies.

Detection of horizontal planar surfaces was reliable. However, surface avoidance of vertical planar surfaces was extremely unreliable. Only two successful trials (defined by no premature triggering of the avoidance behaviour and correct avoidance when presented with a surface) were recorded, out of dozens of such trials.

3.5 Sensing other drones

The Mormoops prototype was always reliably detected by the Crazyflie observer, and both distance (Fig. 9) and orientation in the Crazyflie frame of reference (Fig. 10) could readily be identified. The reverse scenario was not the case, and no reliable estimates of the position of the Crazyflie by the Mormoops observer could be obtained due to signal being overwhelmed by self-noise. Adjustment of gains and more sophisticated filtering may improve detection capability.

The bearing to a secondary drone was always reliably extracted, with a standard error around 30 degrees. The distances were somewhat reliable out to around 20 cm, with a resolution of around 7cm at this distance. Resolution improved as the target drone got closer. At distances greater than one meter, estimates of distance could no longer be obtained with accuracy.

Our Mormoops prototype can thus be detected reliably and consistently from a Crazyflie observer when tethered. The reverse was not the case, largely because the Crazyflie is approximately two orders of magnitude quieter than the pro-

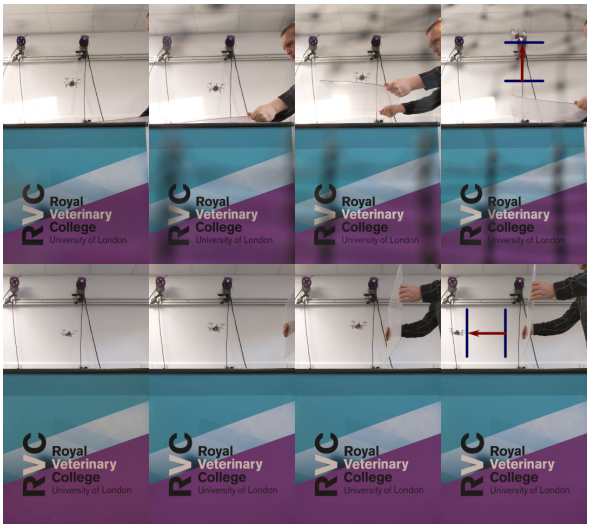


Figure 8: Still images from movie of surface avoidance behaviour. Above: horizontal planar surface avoidance as a surface approaches. Below: wall avoidance as a vertical planar surface approaches.

totype. No signal was therefore detected for our microphones when gains were configured to avoid clipping on the prototype. In short, the configuration for obstacle avoidance on the Mormoops was not compatible with detection of a secondray, much quieter, drone.

4 CONCLUSION

Detection of horizontal planar surfaces is very reliable and robust across modes, tethered, post free flight recording and in drones in flight engaging in surface avoidance. Detection of vertical planar surfaces is reliable while tethered, but far less reliable in the other testing modes. The detection of a larger, louder, more powerful drone from a smaller drone while tethered is possible, and the angle and distance of this drone can be estimated somewhat reliably via acoustic recordings. Detection of the louder drone might be improved by the implementation of motor denoising techniques[23].

For free flight surface avoidance, latency was an issue due to the long chain between sensor systems and flight control (recording to on board microcontroller, to flight control board, to radio, to flight control computer). A reduction in the latency from detection to avoidance could see this evolve into a practical system. At present the latency is around 250ms. Larger spikes in latency are also seen, and arguably the most pressing challenge for further use of the wall detection method is variability in latency as wall effect quickly renders a drone in an unrecoverable state. Reducing latency is achievable and would substantially improve drone sense-and-avoid performance.

These trials were conducted under good conditions, indoors and free from wind, both of which could negatively

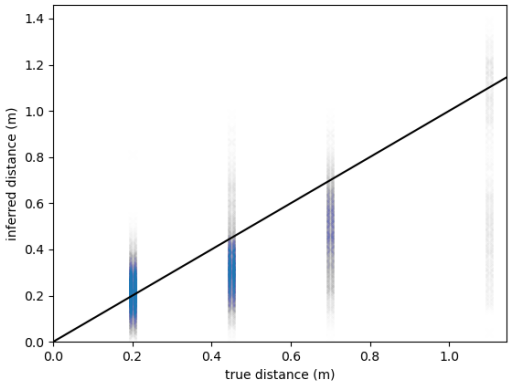


Figure 9: Performance of our machine learning approach when estimating the distance of a target drone from the sensing drone. The x-axis shows the actual distance (ground truth); the y-axis shows the estimate inferred from the microphone recordings and machine learning algorithm. Performance is reasonable for shorter distances, with the error at 20cm approximately 7cm. Shorter range recordings (not shown), suggest accuracy is improved as distance decreases. Above $20cm$ accuracy deteriorates with a bias towards underestimates although estimates of distance are possible and may be useful.

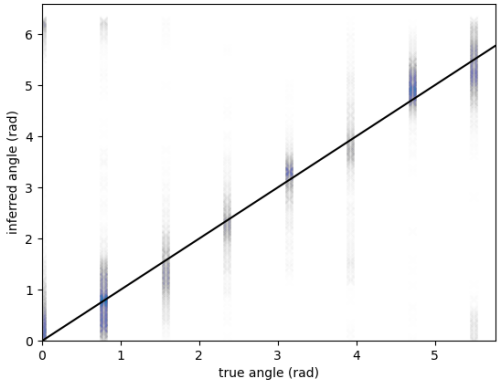


Figure 10: Performance of our machine learning approach when estimating the orientation of a target drone (angle relative to forward on the observer drone). The x-axis shows the actual orientation (ground truth); the y-axis shows the estimate inferred from the microphone recordings and machine learning algorithm. Performance is excellent, with a typical error around 30 degrees, although some variation is observed by angle.

http://www.imavs.org/

contribute to performance of both surface detection and avoidance. It is unclear how robust a neural network based approach will be to the additional sounds that occur outdoors, including wind. Unlike physics based approaches which have predictable responses to new conditions (and which allow for those new condition to be incorporated into the physical model), neural network based approaches may not perform stably in conditions that are not in the training data. One approach to this problem would be to intentionally create noise in the training data to simulate wind or external noise in an effort to make the algorithm more robust. Wind would also likely make avoidance manoeuvres more difficult.

The multiplatform capability of our sensor, and sensor integration, packages mean that multi-drone operation with drone identification are achievable with the current system. We have high temporal resolution digital microphones using the specialised audio processing available on the Teensy should frequency range and audio processing be a limiting factor and we have analogue microphone arrays should spatial resolution and multi-direction sampling be critical.

A quiet, light drones capable of supporting these sensors like the Crazyflie could be used for future development should background motor noise be an issue, and a larger, more capable, but louder, like our Mormoops platform could be employed should extra sensors and equipment be needed for drone identification. This capability could be complimented by integrating acoustic surface detection with surface detection using other sensor modalities such as optical flow or pressure sensors.

This work demonstrates novel capabilities of surface detection via self-noise can be extended in exciting new directions including wider sensor integration and in-flight drone detection.

http://www.imavs.org/

ACKNOWLEDGEMENTS

This work was supported principally by the Defence Science and Technology Laboratory (grant DSTLX-1000143653 to RJB) and, in part, by AFOSR European Office for Aerospace Research and Development (FA9550-19-1-7040 to RJB). We would like to thank Dr. Geoffrey Barrows of Centeye Inc. for useful conversations about drone design and configuration. We would like to thank William Wilson for his efforts in data collection. We would also like to thank the team at the Naval Research Laboratory, particularly Dr. Alisha Sharma, for useful conversations and feedback on the design of neural networks.

REFERENCES

[1] Abhishek Kasturi, Veljko Milanovic, Bryan H Atwood, and James Yang. Uav-borne lidar with mems mirror-based scanning capability. In *Laser Radar Technology and Applications XXI*, volume 9832, pages 206–215. SPIE, 2016.

[2] Joseph Conroy, Gregory Gremillion, Badri Ranganathan, and J Sean Humbert. Implementation of wide-field integration of optic flow for autonomous quadrotor navigation. *Autonomous robots*, 27(3):189–198, 2009.

[3] Toshiyuki Nakata, Nathan Phillips, Patrício Simões, Ian J Russell, Jorn A Cheney, Simon M Walker, and Richard J Bomphrey. Aerodynamic imaging by mosquitoes inspires a surface detector for autonomous flying vehicles. *Science*, 368(6491):634–637, 2020.

[4] Kaman Thapa Magar, Gregory W Reich, Corey Kondash, Keith Slinker, Alexander M Pankonien, Jeffery W Baur, and Brian Smyers. Aerodynamic parameters from distributed heterogeneous cnt hair sensors with a feed-forward neural network. *Bioinspiration & biomimetics*, 11(6):066006, 2016.

[5] Keith Slinker, Corey Kondash, Benjamin T Dickinson, and Jeffery W Baur. High-bandwidth and sensitive air flow sensing based on resonance properties of cnt-on-fiber hairs. *C*, 3(1):6, 2017.

[6] Meysam Basiri, Felix Schill, Dario Floreano, and Pedro Lima. Audio-based relative positioning system for multiple micro air vehicle systems. Technical report, 2013.

[7] Meysam Basiri, Felix Schill, Pedro Lima, and Dario Floreano. On-board relative bearing estimation for teams of drones using sound. *IEEE Robotics and Automation letters*, 1(2):820–827, 2016.

[8] Meysam Basiri, Felix Schill, Dario Floreano, and Pedro U Lima. Audio-based localization for swarms of micro air vehicles. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 4729–4734. IEEE, 2014.

[9] E Tijs, GCHE de Croon, J Wind, B Remes, C De Wagter, HE de Bree, and R Ruijsink. Hear-and-avoid for micro air vehicles. In *Proceedings of the International Micro Air Vehicle Conference and Competitions (IMAV), Braunschweig, Germany*, volume 69, 2010.

[10] Brendan Harvey and Siu O'Young. Acoustic detection of a fixed-wing uav. *Drones*, 2(1):4, 2018.

[11] DA Waters. Echolocation in air: biological systems, technical challenges, and transducer design. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 221(10):1165–1175, 2007.

[12] Jong Hwan Lim and John J. Leonard. Mobile robot relocation from echolocation constraints. *IEEE Transactions on pattern analysis and machine intelligence*, 22(9):1035–1041, 2000.

[13] Frederike Dümbgen, Adrien Hoffet, Mihailo Kolundžija, Adam Scholefield, and Martin Vetterli. Blind as a bat: Audible echolocation on small robots. *IEEE Robotics and Automation Letters*, 2022.

[14] Luke Calkins, Joseph Lingevitch, Joe Coffin, Loy McGuire, Jason Geder, Matthew Kelly, Michael M Zavlanos, Donald Sofge, and Daniel M Lofaro. Distance estimation using self-induced noise of an aerial vehicle. *IEEE Robotics and Automation Letters*, 6(2):2807–2813, 2021.

[15] Honeywell, 2011. Patent number EP 2 386 925A1.

[16] Hans-Elias De Bree and Guido De Croon. Acoustic vector sensors on small unmanned air vehicles. *the SMi Unmanned Aircraft Systems*, 2011.

[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[18] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

[19] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[20] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *International conference on artificial neural networks*, pages 92–101. Springer, 2010.

[21] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM J. Control Optim.*, 30(4):838–855, July 1992.

[22] David W Scott. *Multivariate density estimation: theory, practice, and visualization*, volume 383. John Wiley & Sons, 2009.

[23] Patrick Marmaroli, Xavier Falourd, and Hervé Lissek. A uav motor denoising technique to improve localization of surrounding noisy aircrafts: proof of concept for anti-collision systems. In *Acoustics 2012*, 2012.

http://www.imavs.org/

Deep Learning-based Flight Speed Estimation using Thermal Anemometers

Ze Wang*[1], Jingang Qu*[1,2], Pascal Morin[1] †
[1]Institut des Systemes Intelligents et de Robotique - ISIR
Sorbonne Universite, CNRS, UMR 7222, 75005 Paris, France
[2] IFPEN, France

http://www.imavs.org/

ABSTRACT

This article concerns flight speed estimation from airflow measurements provided by a set of thermal anemometers. Our approach relies on a Gated Recurrent Unit (GRU) based deep learning approach to extract deep features from noisy and turbulent measurement signals of tri-axial thermal anemometers, in order to establish the underlying mapping between the airflow measurement and the flight speed. The proposed solution is validated on a multi-rotor micro aerial vehicle (MAV). The results show that the GRU-based model can effectively extract noise features and perform denoising, and compensate for induced velocity effects along the propellers' rotation axis. As a consequence, robust prediction of the flight speed is performed, including during takeoff and landing that induce ground effects and strong variations of vertical airflow.

SUPPLEMENTARY MATERIAL

Our open source implementation and more details are available at `https://github.com/SyRoCo-ISIR/Flight-Speed-Estimation-Airflow.git`

1 INTRODUCTION

Estimation of the 3D velocity and/or position of a MAV is essential for both tele-operated and fully autonomous missions. Different sensing modalities can be used to this purpose, which can be roughly divided into three categories: *External perception systems* (UWB odometry, GPS[1], Motion Capture system[2]), *Exteroceptive sensors* (Lidar odometry[3], visual odometry(VO)[4, 5]), and *Proprioceptive sensors* (inertial navigation system(INS)[1]). Each category of sensors comes with its own advantages and drawbacks, especially in the context of MAV applications. External perception systems, when available, are very convenient as they require limited payload (both material and computational) aboard the drone, but they are not always available.

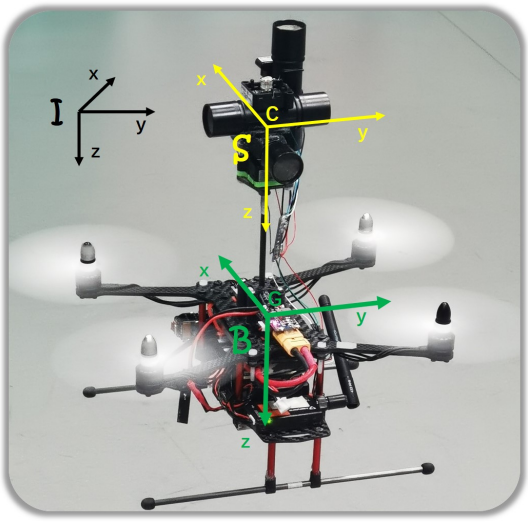


Figure 1: MAV equipped with three thermal anemometers orthogonal to each other.

Exteroceptive sensors provide information on the MAV's environment that is extremely useful for sense and avoid applications, but they often require heavy computational payload, and they are subjected to environmental conditions (light, visual texture, etc). Proprioceptive sensors (INS, or odometry based on Blade Element and Momentum theory for multi-rotor MAVs [6]) use little embarked payload, are always available, and can provide good speed estimates over limited time-intervals, but they are subjected to drift when used for position estimation. Thus, there is no universal sensory modality but each one can be useful or even crucial in specific contexts, thus contributing to a global sensor-fusion architecture.

This paper concerns the problem of velocity/air-velocity estimation from measurements provided by thermal anemometers. Here,"air-velocity" denotes the velocity relative to air. A thermal anemometer, as illustrated on Figure 1, is a cylindrical sensor (this is a typical shape, but not the only one) that provides information about the airflow that passes through the device. It can thus be viewed as a 1D air speed sensor. When used in still air (the typical indoor situation), it thus provides information on the sensor's velocity along the cylinder's axis. When used outdoor, it provides infor-

*The co-authors share the same contribution.

†This work was supported by the French ANR Project DACAR. Email addresses: {firstName.lastName}@sorbonne-universite.fr

http://www.imavs.org/

mation on the sensor's air-velocity along the cylinder's axis, similar to a Pitot tube. Using a set of three orthogonal thermal anemometers to estimate the 3D velocity/air-velocity of a multi-rotor MAV, as illustrated by Figure 1, is not straightforward. The main difficulty comes from the airflow generated by the propellers, especially in the vertical direction (in body frame) where airflow velocity is the sum of the induced velocity and MAV's velocity. In addition, as with any sensor, measurements are noisy, where noise may be electronic or "mechanical" (related, e.g., to vibrations aboard the MAV). By combining a simple analytical sensor model and a neural network architecture, we show that good speed estimates can be obtained in all flight phases, including take-off and landing. In particular, the neural network proves helpful in correcting induced velocity related biases in the (body relative) vertical direction and reducing "noise". Concerning noise reduction, note that similar conclusions about the benefit of neural networks were drawn for attitude estimation based on IMU measurements [7]. We investigated several different neural networks architectures such as LSTM, MLP, and WaveNet, and found that the GRU-based architecture performs best in estimating the flight speed.

The rest of paper is organized as follows. Section 2 provides a literature review concerning the different types of airflow sensors in the context of MAVs applications. A simple model of the thermal anemometer measurement is proposed in Section 3. Outputs of this model are then used as inputs of the deep learning-based architecture presented in Section 4. Experimental results obtained with the proposed method and the MAV of Figure 1 are reported in Section 5. The paper ends with a conclusion and perspectives.

2 AIRFLOW SENSORS FOR MAVS

There are many types of anemometers that rely on different measurement techniques, and they are usually divided into two categories: *velocity anemometers*, which directly convert airflow speed into other physical signals (such as Cup/Vane anemometers, Hot-wire anemometers based on Thermal Dissipation [8, 9, 10], Laser anemometers based on Doppler shift, Ultrasonic anemometers based on ToF[11]), and *pressure anemometers*, which sense pressure differences and thus derive speed according to Bernoulli's principle (such as Pitot-Tube anemometers [12]).

Recently, researchers have also designed custom-made sensors based on the above principles. Inspired by bionic ideas, a whisker-like sensor is presented in [13], and combined with a barometer that acts as a force sensor to measure the airflow pressure signal coming from the whiskers. The idea of a bionic whisker is also employed in [14] to design a dedicated anemometer but Hall sensors are used to measure the displacement of magnetic objects caused by air drag. Another work that also uses Hall sensors is [15], but differently it leverages airflow pressure to push a pendulum-like plate, thus using Hall sensors to measure the angle of deflection and thus predict the airflow speed. Other solutions based on converting the airflow information into a motion information include [13, 14, 15], where both [14] and [15] use Hall sensors to directly capture the electrical signal from the motion, while [13] converts it into a pressure variation on an embedded barometer. In comparison, the whisker-like structure has a smaller size and thus is more sensitive to the variation of airflow and thus has a higher resolution, while the plate structure has a larger measurement range and less turbulence effects. The work [14] stands perfectly in-between: wide measuring range on the one hand and higher resolution on the other hand. All these solutions rely and mechanical moving parts, however, which can be an issue in term of mechanical robustness and maintenance.

Solutions that do not require mechanical moving parts include laser Doppler anemometers, ultrasonic anemometers, thermal anemometers, and Pitot tubes. Laser Doppler anemometers are suitable for high precision scenarios but are not suited to MAVs due to their size and cost. Pitot tubes are unidirectional and usually applied to fixed-wing aircraft. The TriSonica Mini ultrasound anemometer was used in [11]. Although being the most compact 2D ultrasound anemometer on the market, it is still heavy for MAV applications and expensive. The SFM3000 thermal anemometer was used in both [10] and [9]. It is a 1D bidirectional anemometer weighing 17g, with a data rate of 2kHz, a measurement range of 200 standard liters per minute (slm) and a resolution of ±2.5%. This sensor was also chosen for this work due to its numerous advantages: low pressure loss, large flow range, high accuracy, high repeatability, high reliability, good linearity and sensitivity, and easy maintenance due to the absence of moving mechanical parts.

3 THERMAL ANEMOMETER MEASUREMENT MODEL

Bi-directional thermal anemometers[16, 17], measure the airflow speed by monitoring the amount of heat dissipated from a surface using one or more simple temperature sensors. For such a sensor, the 1D bi-directional measurement $m \in \mathbb{R}$ depends on the relative air velocity $v_a \in \mathbb{R}$ along the measurement axis of the sensor, where v_a is the difference between the sensor's linear velocity and the ambient wind velocity, both projected along the measurement axis of the sensor. We model the measurement by assuming a Gaussian stochastic white noise.

$$m = f(v_a) + \varepsilon \ , \quad \varepsilon \sim N(0, \sigma^2) \tag{1}$$

where $f(\cdot)$ is a strictly monotonic function whose inverse function is $g(\cdot)$ and ε is a noise which depends on the sensor characteristics and the turbulence.

3.1 Sliding window average denoising

Thermal anemometers measurement noise usually depends on the characteristics of the sensor itself and the complex turbulence around the sensor. Usually, the circuit noise

and vibration has homoscedasticity characteristics (assumption of equal or similar variances over the entire period) and the complex turbulence can also be regarded as having local homoscedasticity, (e.g. Figure 3, Figure 5). For such signals, sliding window average denoising method is the most easily used to filter out noise. Consequently, the noise-filtered sensor measurement model can be written

$$\hat{m} = f(v_a) \tag{2}$$

3.2 Linear model

A commercially available anemometer usually has a reasonable measurement linearity, so we consider the approximation function as a biased linear function.

$$v_a = \frac{\hat{m} - b}{s} \in \mathbb{R} \tag{3}$$

where b is the bias and s is the scale, which need to be identified from the filtered anemometer measurements $\hat{m}$ and ground truth measurements of v_a. After combining many sampling data, we compose them in the form of a matrix:

$$[v_a] = AX(\hat{m}), \quad A = (-b/s, 1/s) \tag{4}$$

where $[v_a] = (v_a(t_1) \ \cdots \ v_a(t_N))$ and

$$X(\hat{m}) = \begin{pmatrix} 1 & \cdots & 1 \\ \hat{m}(t_1) & \cdots & \hat{m}(t_N) \end{pmatrix}$$

contain ground truth air velocity measurements and anemometer measurements at different sampling times $t_1, \cdots, t_N$, respectively. Ground truth air velocity measurements are provided by a SLAM system, as discussed later in Section 5. Minimizing the squared sum of estimation errors

$$\|[v_a] - AX(\hat{m}))\|^2 \tag{5}$$

gives the least squares estimator

$$\hat{A} = [v_a]X(\hat{m})^T(X(\hat{m})X(\hat{m})^T)^{-1} \tag{6}$$

After identification of A, the air velocity estimate associated with a measurement m is thus given by: $\hat{v}_a = \hat{g}(m) = \hat{A}(1, m)^T$.

3.3 3D triaxial anemometer sensor

In this work, we built a triaxial sensor composed of three anemometers mounted in an orthogonal configuration, as shown on Figure 1. The objective is to estimate the 3D velocity/air-velocity. The following notation is used:

- $\mathcal{I}$ is an inertial frame.
- $\mathcal{B}$ is a body frame (i.e., attached to the MAV), with center at the MAV's center of mass G. The rotation matrix from $\mathcal{B}$ to $\mathcal{I}$ is denoted as R and it satisfies the equation $\dot{R} = R[\omega]_\times$ with ω the angular velocity vector expressed in body frame and $[\omega]_\times$ the skew-symmetric matrix associated with the cross product by ω.
- $\mathcal{S}$ is a sensor frame (i.e., attached to the triaxial anemometer sensor), with center at the sensor's center C. The constant rotation matrix from $\mathcal{S}$ to $\mathcal{B}$ is denoted as R_0 and the vector of coordinates of C in the body frame $\mathcal{B}$ is denoted as δ.
- V^G denotes the coordinates, in inertial frame, of the velocity vector of G (i.e., the MAV's velocity vector).
- V_w denotes the coordinates, in inertial frame, of the wind speed vector.

The objective is to estimate the MAV's air-velocity vector $V_a^G = V^G - V_w$.

Let V^C denote the coordinates, in inertial frame, of the velocity vector of C. We have $V^C = V^G + R(\omega \times \delta)$ and thus

$$V_a^C := V^C - V_w = V_a^G + R(\omega \times \delta)$$

where V_a^C denotes the air-velocity vector at the sensor center C, expressed in inertial frame. Using the linear model proposed in Section 3.2, the triaxial anemometer sensor provides the components of this vector in the sensor frame $\mathcal{S}$. Since the rotation matrix from sensor frame to inertial frame is $\bar{R} = RR_0$, the vector of sensor measurements is

$$\begin{aligned} \bar{V}_a^C = \bar{R}^T V_a^C &= \bar{R}^T V_a^G + \bar{R}^T R(\omega \times \delta) \\ &= R_0^T \left(R^T V_a^G + \omega \times \delta\right) \end{aligned}$$

which is equivalent to

$$R^T V_a^G = R_0 \bar{V}_a^C - \omega \times \delta \tag{7}$$

In the following section we propose a deep learning-based solution to estimate $R^T V_a^G$ from the vector of triaxial sensor measurements $\bar{V}_a^C$ and the gyroscope measurement ω. The constant parameters R_0 and δ in the above relation will be implicitly estimated by the neural network. For the MAV used in this work, R_0 is probably close to the identity matrix but no calibration was performed.

4 DEEP LEARNING-BASED PREDICTION WITH AIRFLOW

Model-based and data-driven solutions are two different state-estimation methods, each of which has different advantages. The model-based approach usually employs Kalman filters [11, 15] to fuse multiple sensor data, low-pass filters [10] to reduce noise, or construct dynamical models [2] to extract airflow speed terms by referring to other measurements, respectively. A data regression method is employed in [9], and a deep learning-based method is employed in [18]. Kalman-based filters are typically the most efficient and reliable for fusing multi-sensor data, while deep learning-based methods have better performance for extracting data features, denoising, and data regression. Turbulence causes heteroscedastic disturbance noise (e.g in Figure 3), which calls for data-driven approaches to identify temporary noise

http://www.imavs.org/

http://www.imavs.org/

characteristics. In addition, the induced flow on multi-rotor MAVs leads to regular coupling of measurement data from anemometers in different directions. Therefore in this section we present our deep learning-based method, which is able to learn local noise features and thus perform denoising, compensate for unknown constant parameters (R_0 and δ), and decouple the nonlinear relationship between data by deep learning a large amount of measurement data. We use a neural network composed of 1D convolution layers and Gated Recurrent Unit (GRU) layers [19].

4.1 Inputs & Outputs

Due to the difficulty to obtain ground truth air velocity measurements, experiments are performed indoor, under the assumption of no ambient wind. Thus the MAV's velocity is equal to its air velocity. The velocity ground truth is obtained from a Visual-SLAM. According to (7), the inputs consist of the air velocity deduced from the raw anemometer measurements and the linear model of Section 3.2, and the angular velocity measurements. Note that we have tried to add accelerometer measurements and motor commands as input of the neural network, but it has led to over-fitting. Inputs (resp. outputs) of the network are expressed in the sensor frame (resp. body frame) in order to keep the network invariant with respect to the orientation of vehicle. This is important in order to reduce the complexity of the network and the amount of training data.

4.2 Neural Network Architecture

The architecture of the neural network is shown on Figure 2. The "Airflow-sensor data" on this figure correspond to the airspeed estimates obtained from the linear model of Section 3.2 using raw (i.e., non-filtered) anemometer measurements. Note, however, that for the linear model identification, i.e., linear regression, anemometer measurement filtering has been performed. The airflow-sensor data, together with the raw gyroscope measurements, are first processed by the scaling layer, which normalizes the inputs. Then, to denoise the inputs, we use two 1D convolution layers, for which the number of filters is 16, the kernel size is 5, the stride is 1, the activation function is the rectified linear unit (ReLU)[20], and the input and output sequences are of the same length by padding with zeros. Finally, two GRU layers with 16 units are stacked on top of convolution layers to handle sequences. Compared to Long-Short Term Memory (LSTM) [21], GRU has fewer learnable parameters and thus less risk of over-fitting. Eventually, a fully connected output layer is used to integrate the deeply processed information to produce the final flight speed prediction. The neural network is built upon TensorFlow[22].

5 IMPLEMENTATION & RESULTS

5.1 Implementation Details

We collect data on the custom-built MAV flight platform of Figure 1 with length x width x height equal to 15x20x20cm and weight about 500g. The MAV is equipped with Pixhawk 4 open source flight control board (STM32f7), D435i binocular depth camera, and Khadas Vim 3 SBC (6-core ARM 2.4GHz).

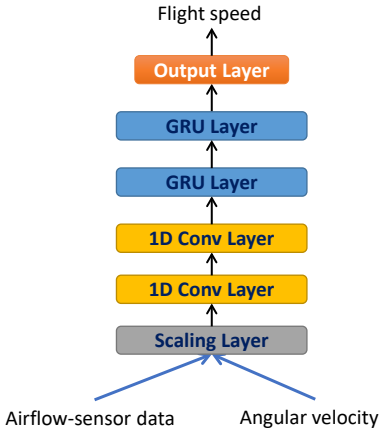


Figure 2: Neural network architecture

The open source flight control system PX4 runs on Pixhawk 4 and contains a 4-loops cascade PID controller, an Extended Kalman Filter (EKF) based state estimator, and uses the mavlink protocol to communicate with Vim 3 via the serial port. All of the components run in a ROS system on Vim 3 and communicate with an off-board laptop via wifi 5. The ground truth velocity is provided by the EKF based on the ORB-SLAM3[5] output and the IMU measurements. The EKF relies on the accelerometer measurement model.

The thermal mass flow sensors we use are the SFM3000-200 from Sensirion AG. The sampling rate of 200Hz here used and the high speed I2C communication interface make it easy to acquire data. All of our flight data has been collected in manual flight or automatic trajectory tracking in a standard gymnasium on different days.

A trajectory generator ROS node commands the vehicle to follow a trajectory in flight, including a circular spiral and a figure-of-8 spiral with a radius of 2 meters and an undulation height of 1 meter. The takeoff operation is also commanded by the ROS node during the automatic trajectory tracking.

The low priority of the data logging, due to the limitations of the autopilot hardware, would lose data randomly at different moments, but the total lost data would not exceed 2% of the total. For this reason, we compensated for the missing ground truth data using a sliding window average cubic polynomial fitting method, while we approximated the missing raw sensor measurements using a linear interpolation method. We do not have statistics of slam accuracy, but the comparison [4] shows that orb-slam3 exhibits performances similar to other state-of-the-art slam methods.

5.2 Neural network training

We divide the collected data into the training set (four flights data with a total of 20.18 minutes), validation set (two

Table 1: Parameters identification with linear regression

Parameters	Sensor-F	Sensor-R	Sensor-D
Bias b	-0.0020	0.0154	-0.2143
Scale s	0.3100	0.2984	0.1420

flights data with 14.47 minutes), and test set (one flight data with 4.04 minutes). To compensate for the differences in flights data on different days (air temperature, humidity, pressure, etc. on the day), both the training data-set and the validation data-set contain multiple flight data from different days. The batches of sequences are created by sliding a 2.5 seconds time window which contains 500 data. The batch size has been set as 512, and we use the Adam with Triangular Cyclic Learning Rate (CLR)[23] as the optimizer, which periodically increases and decreases the learning rate during the training. Early stopping is also used to avoid over-fitting.

5.3 Anemometer measurement noise characteristics

Figure 3 displays air-velocity estimates along the sensor front direction obtained with the linear model, before filtering (blue) and after filtering (red). The flight is composed of five different flight phases performed in automatic flight mode: takeoff, circular spiral trajectory, figure-of-eight spiral trajectory, vertical flight, and landing. The trajectory plots are shown in Figure 4, where the blue curve is the trajectory and the gray curves are its projection onto the x, y, and z axes. We have used here a sliding window of length 19 to generate delay-free smooth data and obtains better noise characteristics. Of course, this sliding window is not used online for the airspeed estimation. The error statistics are presented in Figure 5, which shows that the anemometer measurements present a Gaussian white noise characteristic in the whole and Figure 3 shows that the measurements locally present a disturbance characteristic related to the flight state.

5.4 Prediction of linear model and GRU model

For the test flight, the trajectory of Figure 4 was used. The prediction results are shown in Figure 6, where we compare the flight speed prediction of the linear model of Section 3.2 (green dashed line) with the flight speed prediction of the GRU model (blue dashed line) and the ground truth flight speed (red solid line).

The parameters of the identified linear model are presented in Table.1, where Sensor-F, Sensor-R, and Sensor-D stand for the Front, Right, Down directions in the sensor frame respectively. As expected from the symmetry of the system, parameters for the Sensor-F and Sensor-R are very close. Concerning the vertical direction (i.e., Sensor-D), we identified this model using only flight data from hovering and vertical flights, so as to avoid the impact of horizontal flight velocity on the induced velocity. When the horizontal flight is dominant, the linear model estimation in the F-R direction shows essentially the consistency with the ground truth, while the resulting induced flow causes bias in the D-direction estimation. When vertical flight is dominant, the air is more likely to be accelerated downward in comparison with the drone's body and thereby change the induced flow speed faster than flight speed, so that the linear model estimation in D-direction is ahead of the true flight speed. The vertical flight speed also shows a slight coupling characteristic in the horizontal measurements.

Figure 7 shows the flight speed prediction error of the linear model w.r.t the ground truth versus the flight speed prediction error of the GRU model w.r.t the ground truth in each direction. The variances of the GRU model in each direction are 0.0111, 0.0132, 0.0143, while that of the linear model in the F-R direction are 0.0421, 0.0377, respectively.

We summarize as follows: (1) The prediction results of GRU largely overlap with the ground truth, better than linear model, which indicates that the neural network successfully discovers patterns in the data. (2) The GRU model flight speed prediction curve is smooth, which indicates that the neural network identifies the noisy feature and thus performs denoising process. (3) The anemometer measurements in the D-direction implicitly contain the induced flow speed, which is coupled with the flight state, and which is usually difficult to predict and the linear model performs poorly for the prediction of the flight speed in trajectory tracking. The GRU model, however, shows better performance, especially during the takeoff and landing phases. This implies that the GRU model is very robust. (4) For the flight speed prediction in the F-R direction, both the linear model and the GRU model match the ground truth well but Figure 7 shows that the GRU model largely outperforms the linear model.

6 CONCLUSIONS & FUTURE WORK

In this work, we addressed the velocity/air velocity estimation problem aboard MAVs based on thermal anemometer measurements. The proposed solution uses a set of three sensors mounted in a triaxial orthogonal configuration and combines a simple analytical model with a neural network. By performing all computations in body and sensor frames, the problem is invariant with respect to rotations, which reduces the complexity of the neural network and the amount of required training data. Validations show that the neural network brings significant benefits in term of denoising and compensation of induced-flow.

Perspectives include using other sensors available onboard the MAV, such as accelerometer barometer, to further improve speed prediction by using a Kalman method, or exploiting anemometer measurements for obstacles detection.

ACKNOWLEDGMENTS

The first author is funded by the China Scholarship Council (CSC) from the Ministry of Education of P.R. China. Thank to Prometheus platform [24] support.

http://www.imavs.org/

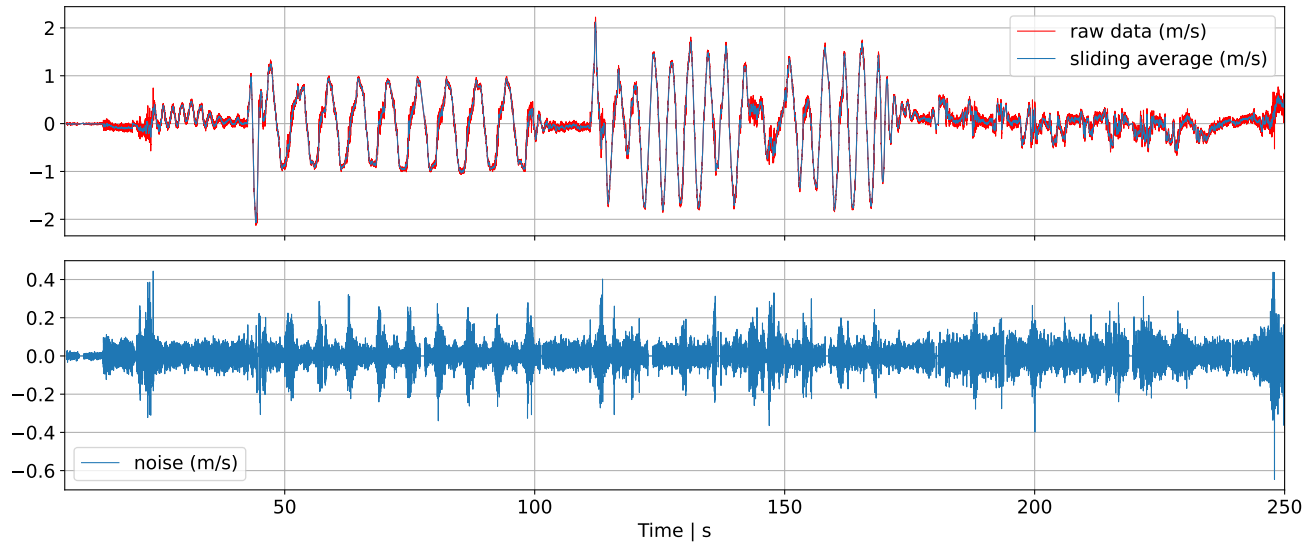


Figure 3: Sliding window average denoise. [Top] The air-velocity estimate with the linear model in the Sensor Front direction before filtering (blue) and after filtering (red). [Bottom] The filtered noise shows that it presents almost homoscedasticity over the entire period, while locally, especially when the flight speed direction changes, it presents heteroscedasticity from the whole.

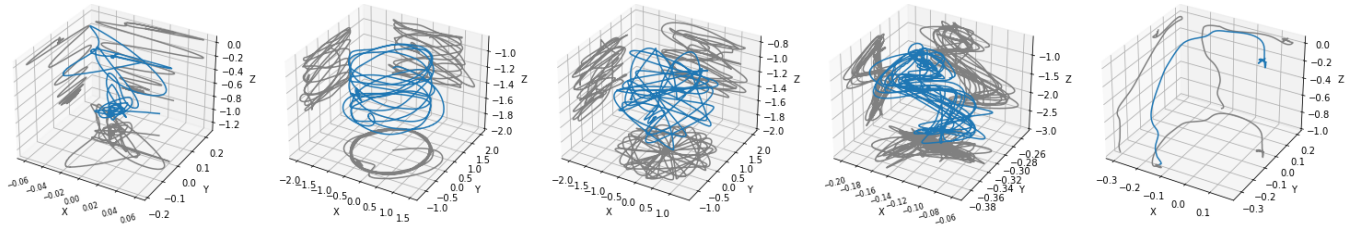


Figure 4: The plots show takeoff, circle spiral, figure-of-8, vertical flight, and landing in order from left to right.

http://www.imavs.org/

References

[1] AH Mohamed and KP Schwarz. Adaptive kalman filtering for ins/gps. *Journal of geodesy*, 73(4):193–203, 1999.

[2] LNC Sikkel, Guido CHE de Croon, Christophe De Wagter, and QP Chu. A novel online model-based wind estimation approach for quadrotor micro air vehicles using low cost mems imus. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2141–2146. IEEE, 2016.

[3] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, pages 1–9. Berkeley, CA, 2014.

[4] Dinar Sharafutdinov, Mark Griguletskii, Pavel Kopanev, Mikhail Kurenkov, Gonzalo Ferrer, Aleksey Burkov, Aleksei Gonnochenko, and Dzmitry Tsetserukou. Comparison of modern open-source visual slam approaches. *arXiv preprint arXiv:2108.01654*, 2021.

[5] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[6] Alexandre Letalenet and Pascal Morin. Identification and evaluation of a force model for multirotor uavs. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4280–4286. IEEE, 2020.

[7] Martin Brossard, Silvere Bonnabel, and Axel Barrau. Denoising imu gyroscopes with deep learning for open-loop attitude estimation. *IEEE Robotics and Automation Letters*, 5(3):4796–4803, 2020.

[8] Paolo Bruschi, Massimo Piotto, F Dell'Agnello, J Ware, and N Roy. Wind speed and direction detection by means of solid-state anemometers embedded on small quadcopters. *Procedia Engineering*, 168:802–805, 2016.

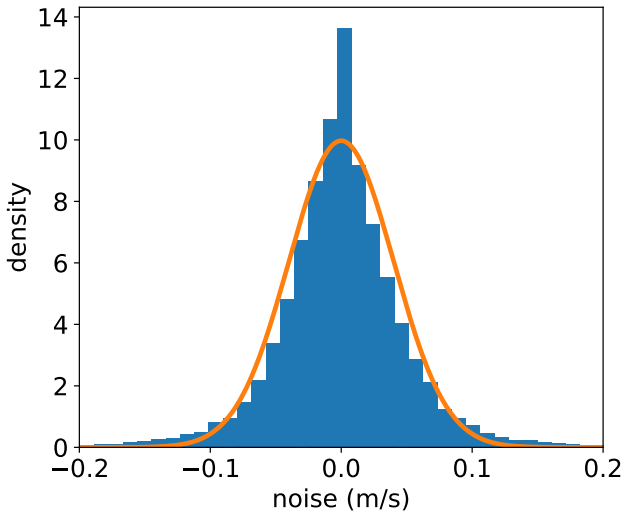


Figure 5: Histogram of the statistical distribution of the filtered noise. Generally, the noise presents Gaussian white noise characteristics with the mean of $2.5 \times 10^{-3} m/s$ and the std of $0.04 m/s$.

[9] Jörg Müller, Oliver Paul, and Wolfram Burgard. Probabilistic velocity estimation for autonomous miniature airships using thermal air flow sensors. In *2012 IEEE International Conference on Robotics and Automation*, pages 39–44. IEEE, 2012.

[10] You Li, Shady Zahran, Yuan Zhuang, Zhouzheng Gao, Yiran Luo, Zhe He, Ling Pei, Ruizhi Chen, and Naser El-Sheimy. Imu/magnetometer/barometer/mass-flow sensor integrated indoor quadrotor uav localization with robust velocity updates. *Remote Sensing*, 11(7):838, 2019.

[11] Derek Hollenbeck, Gregorio Nunez, Lance E Christensen, and YangQuan Chen. Wind measurement and estimation with small unmanned aerial systems (suas) using on-board mini ultrasonic anemometers. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 285–292. IEEE, 2018.

[12] Sam Prudden, Alex Fisher, Matthew Marino, Abdulghani Mohamed, Simon Watkins, and Graham Wild. Measuring wind with small unmanned aircraft systems. *Journal of Wind Engineering and Industrial Aerodynamics*, 176:197–210, 2018.

[13] William Deer and Pauline EI Pounds. Lightweight whiskers for contact, pre-contact, and fluid velocity sensing. *IEEE Robotics and Automation Letters*, 4(2):1978–1984, 2019.

[14] Andrea Tagliabue, Aleix Paris, Suhan Kim, Regan Kubicek, Sarah Bergbreiter, and Jonathan P How. Touch the wind: Simultaneous airflow, drag and interaction sensing on a multirotor. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1645–1652. IEEE, 2020.

[15] Shady Zahran, Adel M Moussa, Abu B Sesay, and Naser El-Sheimy. A new velocity meter based on hall effect sensors for uav indoor navigation. *IEEE Sensors Journal*, 19(8):3067–3076, 2018.

[16] Oleg Sazhin. Novel mass air flow meter for automobile industry based on thermal flow microsensor. i. analytical model and microsensor. *Flow Measurement and Instrumentation*, 30:60–65, 2013.

[17] Doug Sparks. Mems pressure and flow sensors for automotive engine management and aerospace applications. *MEMS for Automotive and Aerospace Applications*, pages 78–105, 2013.

[18] Andrea Tagliabue and Jonathan P How. Airflow-inertial odometry for resilient state estimation on multirotors. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5736–5743. IEEE, 2021.

[19] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[20] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[22] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

[23] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017.

[24] Prometheus Autonomous drone open source project. `https://github.com/amov-lab/Prometheus/`.

http://www.imavs.org/

http://www.imavs.org/

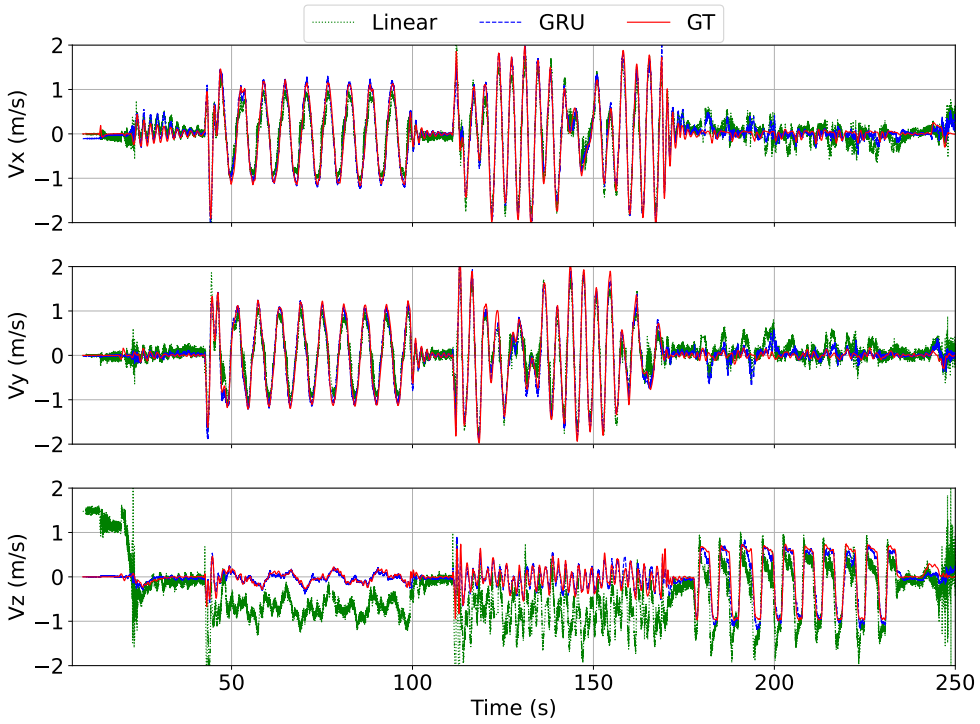


Figure 6: Five flight phases are respectively in chronological order the take-off (<30s), the circular spiral trajectory (40s-100s), figure-of-eight trajectory (120s-170s), vertical flight (180s-230s), and the landing (240s-250s). The flight speed prediction of the identified linear model without removing the noise (green) vs. that of the learning-based model (blue) vs. the ground truth flight speed from V-SLAM (red). All is expressed in Body-FRD frame.

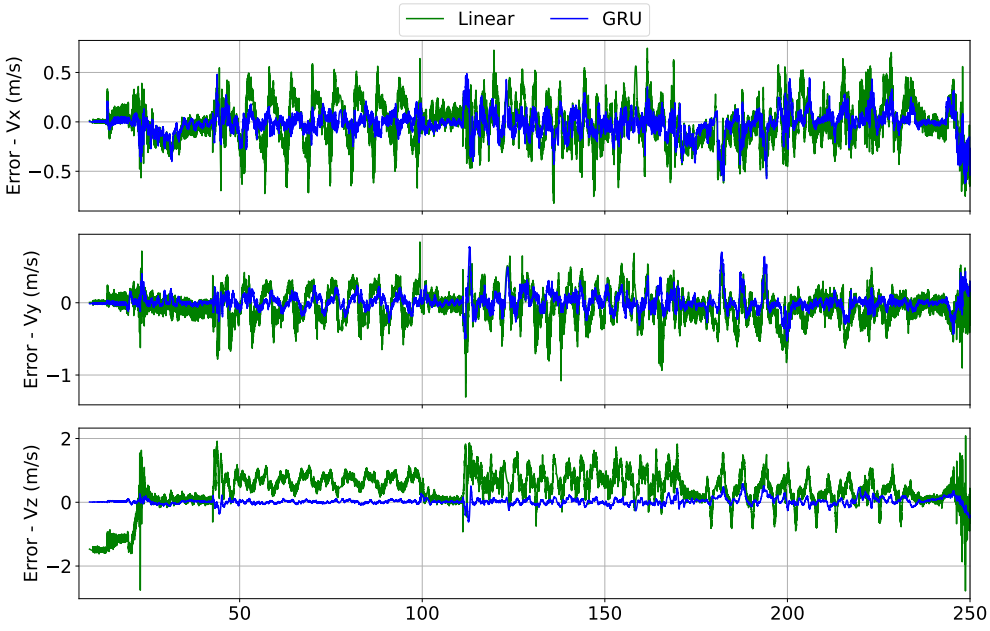


Figure 7: The flight speed prediction error of linear model w.r.t the ground truth (green) vs. the flight speed prediction error of deep learning-based model w.r.t the ground truth (blue). The variances of the GRU model in the each direction are 0.0134, 0.0133, 0.0139, while the variances of the linear model in the F-R direction are 0.0421, 0.0377, respectively.

Semantic Segmentation of RGB Aerial Images Using CNN and Superregions

Mariana-Edith Miranda-Varela, L. Oyuki Rojas-Perez, and Jose Martinez-Carranza*

Instituto Nacional de Astrofisica, Optica y Electronica (INAOE), Puebla, Mexico

http://www.imavs.org/

ABSTRACT

In recent years, large amounts of aerial images are readily available because they are taken from aeroplanes, hot-air balloons, and unmanned aerial vehicles. However, these images are not relevant if their information is not interpreted. Although there are several techniques to analyse them, i.e., from statistical methods to machine learning techniques, at present the most popular technique is deep learning, where the convolutional neural network (CNN) has shown good accuracy and success on several studies of satellite images with a ground sampling distance $>$ 10 m. In this work, we employ red-green-blue (RGB) aerial images taken from 100-300 meters above the ground, which are employed to build a CNN based on a variant of a VGG16 architecture, for the semantic image segmentation. Four classes are identified, urban zone, vegetation zone, agricultural zone and roads. Due to the high spatial resolution, pixels are grouped with respect to similar texture in superregions to reduce the number of pixels classified by the CNN. Our approach is able to segment an aerial image in three modalities: i) pixel-wise segmentation; ii) superpixels, this is, a group of pixels; iii) superregions, which is a group of superpixels. The model is tested with aerial images with a height between 200 to 400 meters. According to our results, the accuracy is similar for each one of the three methods presented in this work. However, the time performance is significantly reduced when employing superpixels and superregions. Furthermore, we have obtained an average F-score 0.732, comparable to state-of-the-art approaches.

1 INTRODUCTION

At present, the interpretation of satellite and aerial image is an essential task, because this kind of images provides important data to bring about land cover map, land use map, and object detection. The land cover and land use maps can be used as a tool to plan a city [1] or to monitor either deforest or changes of a particular region [2]. Moreover, this information can be recorded to form a geographical database. Concerning object detection, some types of targets can be vehicles, buildings, roads [3], aeroplanes [4], and so on.

Semantic segmentation is part of visual computation that assigns a classification label to each pixel of an image. There are many algorithms to segment an image, which are classified in: threshold-based, edge-based, region-based and classification-based methods, where the last is able to get more accuracy values with appropriate feature extractors and classifiers [5]. Deep learning models are used to classify or segment an image, where Convolutional Neural Network (CNN) reported the best performance. A CNN is a single model, for this reason it is regarded as a one-step method [5].

To build a CNN model for semantic segmentation is necessary a dataset, which is employed to train and test the model. Some dataset available online are ISPRS benchmark[1], Google Maps, Linz Data Service[2], among others. Generally, these are composed of high-resolution orthophotographs with a ground sampling distance (GSD) of $>$ 10 cm [6]. Most of the studies of semantic segmentation using CNN employ mainly the ISPRS benchmark, that makes up of high-resolution images and the digital surface map (DSM), where the images consist of near infrared (NIR), red and green bands, and they have a GSD of less than 10 cm. The NIR can be used to compute the Normalized Difference Vegetation Index (NDVI) and the Green Normalized Vegetation Index (gNDVI) using the red and the green band, respectively. Both indexes are utilized to distinguish easily the vegetation into an image [7]. Furthermore, DSM provides height information, that improves the precision of the semantic segmentation [8–10].

At present, most of the research on semantic segmentation has focused on producing a CNN model through high-resolution orthophotographs and its DMS. In contrast to, the CNN models build by means RGB images have been less explored. Moreover, in this work, we consider aerial images taken from a lower distance than the high-resolution orthophotographs, that is between 100 and 300 meters above the ground. Thereby these images contain small area in greater detail. As far as we know, there is not approach to build

*Email address(es): carranza@inaoep.mx

[1] http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html

[2] https://data.linz.govt.nz/

a CNN model using RGB aerial images taken close to the ground. The classification classes for our work are roads, agricultural zone, urban zone and vegetation zone.

In this paper, a CNN model is produced through raw RGB aerial images to achieve a semantic segmentation at pixel-wise classification. To take advantage of the high spatial resolution of images, preprocessing techniques to group pixels similar in colour are employed, and, thereby the number of pixels classified is reduced. The first technique yields superpixels, which are grouped based on their average colour to constitute superregions. The structure of the paper is as follows: Section 2 describes the related work. Section 3 details the implemented system. In Section 4 the experimental results are presented and in Section 5 the conclusions and future work are exposed.

2 RELATED WORK

Semantic segmentation is a challenging task for satellite and aerial images, because of the variety of information on an image, such as texture or illuminance. On the one hand, different types of textures for buildings or vegetation are present. On the other hand, the relative absence of light, shadows, are a source of noise to classify an aerial image correctly. There are many techniques to segment this kind of images, although those works that employed a deep learning model have reported the best performance [11].

SegNet has proved to get better results for segmenting images [12]. Thereby, this architecture has been employed in several works for semantic segmentation of satellite and aerial images. [13] extended SegNet architecture to distinguish eight classes: building, sealed area, bare soil, grass, tree, water, car and other on multi-spectral images (RGB + NIR). Besides, DSM and Digital Terrain Model (DTM) were used to represent the height information of the experiment region. In [11], a SegNet variant was proposed to identify roads, buildings, low vegetation, trees and cars on NIRRG images. Apart from the dataset, DSM, Normalized DSM (NDSM), and NDVI were used to improve the final prediction.

One of the most important benchmarks for semantic segmentation of aerial images is ISPRS, which has been employed in the following approaches. [14] proposed a Local Attention Network (LANet), composed by a patch attention module (PAM) and an attention embedding module (AEM), meanwhile, PAM improves the local context information, AEM enhances the use of spatial information. [15] solved it by means a multi-skip convolutional network and Markov Random Fields. The former extracts the context information and the latter refines the results. Although their segmentation is effective, the time required to get it is high. In the aforementioned approaches, the categories identified were: roads (impervious surfaces), building, low vegetation, tree, car and clutter/background. [10] solved the semantic segmentation utilizing CNN and hand-crafted features, in the end, a Conditional Random Field (CRF) was applied. They identified five classes: impervious surfaces, building, low vegetation, tree, and car. Moreover, the pretrained model was tested on another type of data. The best performance was got with NIRRG images, DSM and NDSM. In [16] a Fully Convolutional Networks (FCNs) is proposed, which has two identical CNN that are merged before the final layer. The input for each CNN is the image and the digital elevation model (DEM), respectively. TreeSegNet is proposed in [17], which is composed on three elements: (1) the segmentation module, that is, an encoder-decoder architecture; (2) the Tree-CNN block, which improves the result of the previous element and it is the centre part of this approach, and (3) the concatenating connections. This approach was trained and tested through five channels R, G, B, NIR and DMS. [18] proposed an encoder-decoder architecture with skip connections, and Fully connected CRF (FCRF) post-processing, which improved the results. The input CNN model was the NIRRG images and NDSM from ISPRS benchmark. In [19], a hybrid FCN architecture is proposed, which merge the image channels with DSM. Another approach based on CNN is presented in [9], where CNN was tested on multispectral orthoimageries and DSM to classify vegetation, ground, roads, buildings and water.

According to the results of previous works, semantic segmentation using CNN reports a high level of accuracy, moreover, the use of the DMS improved the performance measures [8–10]. However, DSM is more common for satellite images than aerial images and the NDVI can be computed only in those images with NIR channel. Due to the semantic segmentation utilizing only RGB aerial images has been disregarded, in this work builds a CNN model through RGB aerial images without extra information, such as DMS or NDVI. Moreover, the images of interest are those taken from 100 and 300 meters above the ground.

3 METHODOLOGY

In this section, the elements employed in our approach for segmenting aerial images are presented. Our proposal identifies four classes: agricultural, urban, vegetation and roads. Firstly, the dataset employed to train the model is exposed. Then, CNN architecture is described in detail. Finally, the types of classification are explained.

3.1 Dataset

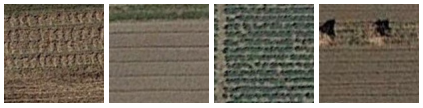
The set of images employed to train the model is composed of 36000 images of size 60×60, named patches, which were extracted from Google Earth images correspond to the Megalopolitan area at the centre of Mexico[3] and pictures of different states of Mexico taken from drones. The altitude of the Google Earth images is between 200 and 300 meters, and aerial images taken by a drone from a range of height of 50

[3]Mexico City, the State of Mexico, Tlaxcala, Hidalgo, Morelos, and Puebla constitute the Megalopolitan area.

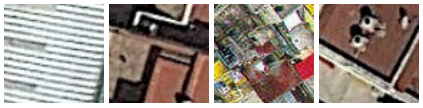
http://www.imavs.org/

and 100 meters.

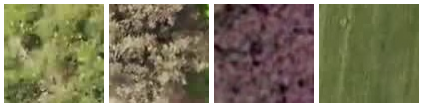
All aerial images were cropped into non-overlapping tiles of size 60×60 pixels, where each one has information corresponding to one class. The total of patches for each class is 9000. In Figure 1, four patches for each class are presented. Agricultural patches present information of sown, harvested and plantations grow land. Concerning urban patches, roof, roofing sheets, and objects that do not belong to the other classes. Information presented in vegetation patches is grass and trees. Finally, road patches contain roads and earthen roads.



(a) Agriculture patches.



(b) Urban patches.



(c) Vegetation patches.



(d) Road patches.

Figure 1: Example of patches for each class of the dataset.

Due to CNN requires a large amount of training data, the dataset was increased by the synthetic data augmentation technique. The applied transformation to the dataset was rotation and translation, with values of 45° and 90, respectively.

3.2 CNN architecture

There are several architectures for deep learning in the literature, however, the CNN have been applied in classification and semantic segmentation approaches with successful results [15]. CNN was inspired by the visual cortex organization of the living creatures [20], and it is composed by one or several CNN building blocks, that is, convolutional layer, activation function and pooling layer, followed by one or more fully connected layers and the loss layer [13].

The features of the input image are extracted by CNN layer through filters, named kernels, which are generally of size 3×3. This layer is the most important of the CNN building, its output is transformed employing an activation function. Following, several features are combined with the pooling layer. In this work the size of kernel used is 3×3 with stride 1. Moreover, the rectified linear unit (ReLU) is employed as the activation function, which preserves only the positive values, and the pooling method employed was max-pooling with a size of 2×2. In Figure 2 the CNN architecture employed is shown, which is based on VGG16 [21]. After the last block building CNN is employed one fully-connected layer, followed by one dropout layer with proportion 0.5, this is employed to prevent the overfitting and, finally, a softmax classifier which computes the probability for each class is added. The size of output vector is four which corresponds to the next classes: urban zone, vegetation zone, agricultural zone and roads. The input to the CNN model is a RGB patch of size 60×60, and the output is the label corresponding to the central pixel which is classified according to the rest pixels around it, this is considered as a patch-based approach. Our CNN model was trained from scratch with random initialization, with 100 epochs, and 80% of dataset were chosen randomly for training and the rest was employed for testing the CNN model. Furthermore, the base learning rate is set to 1E-5 with a batch size of 100.

Our approach can segment an aerial image of any size, however, the minimum size is 60×60. Due to the CNN model predicts the label of central pixel, this proposal is named pixel-based classification. As a consequence, the number of classified pixels is based on the size of the image, multiplying weight × height, which is a disadvantage for large images.

3.3 Superpixel-based classification

The pixel-based classification can produce a salt-and-pepper effect on the output image. This problem was tackled applying a simple segmentation technique, which groups adjacent pixels based on their colour and texture similarity [22], each group is named superpixel. This reduces the complexity of an image owing to the number of superpixels is lower than pixels. Therefore, the time required to segment an image is lower than pixel-based classification approach. Moreover, the appearance of spots is reduced. To extract superpixels of aerial image SLIC algorithm [23] is employed because this algorithm is faster than others superpixels algorithms and it has a good performance. SLIC is based on k-means using color and spatial information [22]. This approach is denoted as superpixel-based classification.

3.4 Superregion-based classification

Due to the lower distance considered for RGB images, can exist several superpixels with similar texture, which can be joined on large areas, denoted as superregion [24], also named super-segments. Therefore, a superregion is a connected set of adjacent superpixels, which have a bit difference of colour, that is computed by Eq. 1.

$$D_c = \sqrt{(r_j - r_i)^2 + (g_j - g_i)^2 + (b_j - b_i)^2} \qquad (1)$$

where D_c represents the Euclidean distance between i-superpixel and j-superpixel, which must be adjacent. r, g,

http://www.imavs.org/

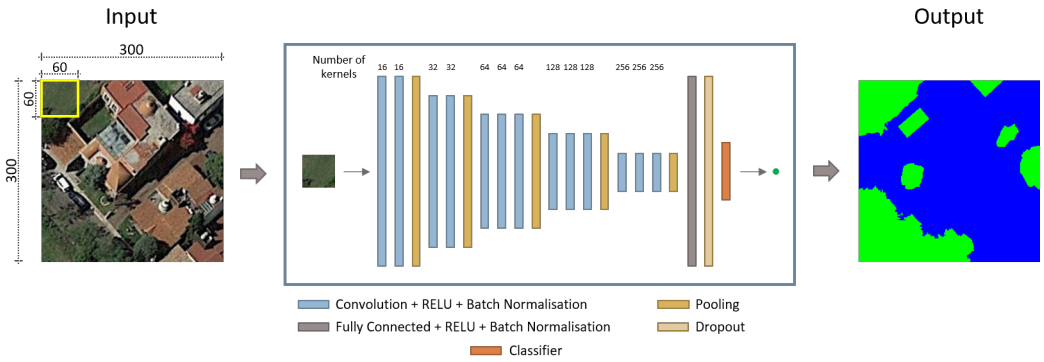


Figure 2: CNN architecture, which is composed by thirteen convolutional layers, one fully connected layer, a dropout layer and a classifier. On the top of each convolutional layer the number of kernels is shown.

and b are average of red, green, and blue values for i and j superpixels. To merge i and j superpixels, the D_c value must be less than a threshold.

An advantage of superregion is the time required to segment an image, which is lower than previous approaches because only a subset of superpixels for each superregion is classified, thereby, the number of superpixels classified is lower than the superpixel-based classification. Furthermore, the salt-and-pepper effect is reduced and the edges for each class are improved.

In Figure 3 an example of the original image for Image 1 (see Table 1), its version dividing into superpixel, where each superpixel is surrounded by a green line, and image dividing into superregion, whither each superregion is represented by different colour are shown. The number of pixels (Figure 3(a)) is 360000, meanwhile the number of superpixels (Figure 3(b)) is 1069 and the number of superregions (Figure 3(c)) is 421.

4 EXPERIMENTS

In this section, the use of our model to segment aerial images is studied. The model was developed with libraries Keras and TensorFlow, and the experiments were conducted on a 2.6 GHz Intel Core i7 processor, RAM of 15GB and Nvidia GeForce GTX 970M GPU. In the following experiments, five aerial images are segmented, which are taken between 400 and 200 meters. The images, their size and their ground truth are presented in Table 1. Regarding ground true, roads are represented by brown colour, the vegetation zone is coloured with green colour, the urban zone is shown by blue colour and the agricultural zone is painted with yellow colour. With respect to Image 3, the architecture of the houses is different with respect to those employed to train the model.

The test images contain city and suburban surfaces, concerning the urban images present mixture textures for roofs, such as concrete, steel, and wooden. Meanwhile, the suburban environments are constituted by agricultural fields and vegetation regions, mainly.

4.1 *Performance measures*

In order to compute the accuracy of our model, three metrics are employed:

$$
\begin{aligned}
Precision &= \frac{TP}{TP+FP} \\
Recall &= \frac{TP}{TP+FN} \\
F\text{-}score &= \frac{2}{\frac{1}{Precision}+\frac{1}{Recall}}
\end{aligned}
$$

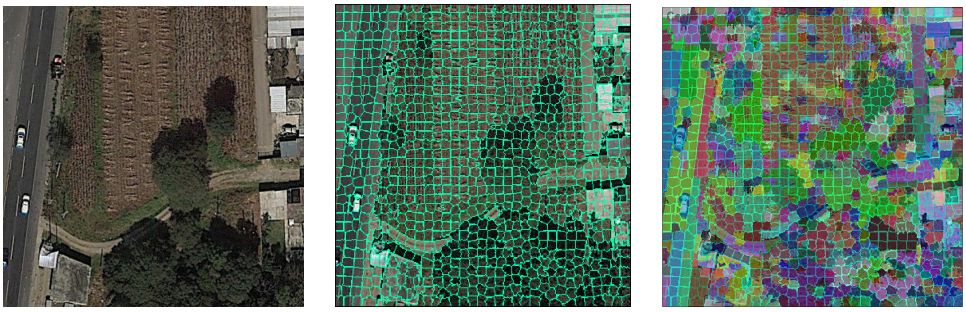
where TP (True Positive) is the number of pixels correctly classified, FP (False Positive) is the number of pixels that according to CNN belong to one class but they really belong to another, and FN (False Negative) is the number of pixels that according to CNN do not belong to a class but they actually do. For each measure, its value is a real number between 0 and 1 and a value close to 1 is preferred. Consequently, $Precision$ is the ratio of the correctly classified pixels to all classified positive pixels, $Recall$ is the ratio of the correctly classified pixels to all actual positive pixels, and $F\text{-}score$, also called $F1\text{-}score$ or F measure, represents the balance between Precision and Recall.

The aim of this paper is to segment RGB aerial images employing a CNN model. For this, we adopted three schemes: pixel-based classification, superpixel-based classification and superregion-based classification, where the last two experiments considered a set of pixels with colour similarity.

For each experiment, the values of precision, recall, and F-score corresponding to each class (road, agricultural zone, urban zone, and vegetation zone) for Image 1 and 2 are computed. Furthermore, the final performance values for all images are shown. In the second experiment, the number of superpixels is calculated by means Eq. 2:

$$no\_spx = width \times height \times 0.003 \qquad (2)$$

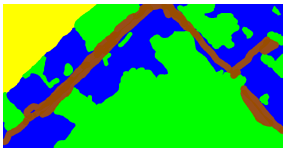
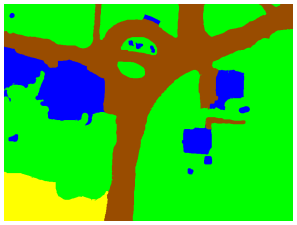
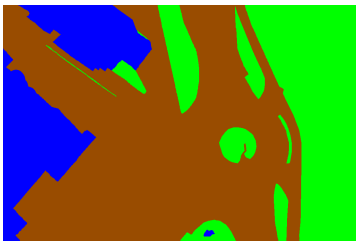
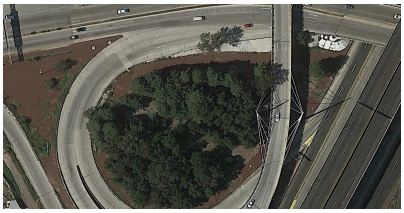
http://www.imavs.org/



(a) (b) (c)

Figure 3: (a) Original image. (b) Image partitioning into superpixels. (c) Image partitioning into superregions.

Table 1: RGB aerial image test, their size and their ground truth. The colours employed in ground true are paths in brown, green zone in green, urban zone in blue and agricultural zone in yellow.

Id / Size	Image	Ground Truth
Image 1 600×600		
Image 2 900×463		
Image 3 900×675		
Image 4 1100×733		
Image 5 1651×850		

http://www.imavs.org/

where width and height correspond to the size of each image, the constant value, 0.003, was defined by a set of experiments. Finally, the third experiment, a superregion is composed of adjacent superpixels whose D_c is less than 30 (see section 3.4). The aim to group a superregion is to take advantage of the colour similarity among a set of adjacent superpixels, and only a subset of a randomly chosen superpixels was classified. For the last experiment, we consider two sizes of subsets, which were a) a quarter, and b) the half of the total superpixels of a superregion. To define the class of a superregion, four counters were considered (one for each class), which were initialized to zero and were increased according to the class assigned by the CNN model. When the evaluation of the subset of superpixels was finished, the class of the superregion was related to the highest counter.

In Table 2, the number of elements classified by our model are listed, where the number of elements is the highest for the first experiment, meanwhile the lowest number corresponds to the superregion-based with a quarter of elements. Corresponding to superpixel-based and superregion-based with half of the elements, there is no substantial saving in the elements classified. Concerning to the time required (in seconds) for preprocessing and to segment an aerial image is presented in Table 3. Wherein the behaviour is similar to the number of elements classifies, so that to high number of elements, the time required is high (pixel-based classification) meanwhile to lowest number of elements, the time required is low (superregion-based classification evaluating a quarter of superpixels). The difference between superpixel-based and superregion-based classification is between 2 and 3 seconds. On the other hand, the time required for grouping pixels in superregions is greater than the time to segment from Image 3 to Image 5, their final time are lower than pixel-based classification.

4.2 Pixel-based classification

In the first experiment, the number of elements classified was defined by the size of the image, that is, $weight \times height$. The performance measures for Image 1 and 2 for each class are shown in Table 4. Although our model was trained with another type of architecture of houses, it was able to classify correctly more than the half of pixels of Image 2 for the urban zone (F-score 0.762618). On the other hand, the worst F-score value was obtained for roads, 0.305186, which was less than 0.802189 (Image 1). The previous result can be generated because the patches corresponding to urban zone and roads in some cases have similar textures, in particular in those roof without waterproofing. The average performance values are presented in the first and the second rows of Table 5.

In Table 5 the average performance measure for each image are presented. According to the average of the precision (0.729796), the recall (0.736097) and the F-score (0.728108) values, our model is able to classify the most of pixels correctly. However, it mistook when the altitude of the image is near the 400 meters (Image 2) and the architecture of the houses is different to those employed to train the CNN model (Image 3).

4.3 Superpixel-based classification

In the second experiment, the number of elements classified was defined by Eq. 2. In Table 6, the performance measures for Image 1 and 2 for each class are presented. According to the F-score values, the use of superpixels improves the semantic segmentation for Image I, however the F-score values for agricultural zone and urban zone are worst than pixel-based classification approach (see Table 4). The average performance values (bold text) are presented in the first and the second rows of Table 6.

The average performance measure for each image using superpixel-based classification are exposed in Table 6. In accordance with the performance values, this approach improves the accuracy for Image 1, 3 and 4, due to the values are higher than those reported in Table 4. Additionally, the average values for this experiment has been improved.

4.4 Superregion-based classification

Due to the random selection of superpixels to define the class for a superregion, in this experiment ten runs were executed, therefore the statistical values (B: best, W: worst, Av: average, Md: median and SD: standard deviation) for precision, recall and F-score measures are presented in Tables 8 and 9. In Tables 10 and 11 the median values for ten runs are shown.

In Table 8, the statistical results for ten runs evaluating half of superpixels for a superregion is presented. For Image 1, the performance is better than the results to Image 2. With respect to the statistical results for ten runs evaluating a quarter of superpixels for a superregion (see Table 9), the performance values are lower than the first, however this semantic segmentation reduce the number of superpixels evaluated, therefore this is more fast.

In Tables 10 and 11 the average median values are presented for half and a quarter of superpixels, respectively. According to these values, the evaluation of half of superpixels reported the best performance measures, meanwhile the evaluation of quarter of superpixels had a similar behaviour than superpixel-based classification.

4.5 Graphical comparison

In Figure 4 the output images corresponding 1 and 2 are presented. Concerning to output Image 1 (middle column), pixel-based classification presents the salt-and-paper effect in the boundary of each region, mainly. Moreover, the regions labeled by a number represent misclassified regions because this are shadows. The regions surrounded by red rectangle represent an area of image with low accuracy of prediction and regions marked off pink rectangles misclassified some pixels. Superpixel-based classification and

http://www.imavs.org/

Table 2: Number of elements classified by our model for each experiment.

Image	Pixel	Superpixel	Superregion	
			Quarter	Half
Image 1	360000	1069	460	960
Image 2	416700	1230	616	1062
Image 3	607500	1769	805	1581
Image 4	806300	2451	720	2020
Image 5	1403350	4127	1338	3722
Total	**3593850**	**10646**	**3939**	**9345**

Table 3: Time in seconds required for preprocessing and to segment each image for each experiment.

Image	Preprocessing		Segment			
	Superpixel	Superregion	Pixel	Superpixel	Superregion	
					Quarter	Half
Image 1	0.4971	3.1863	1448.0335	9.7861	3.8260	4.9720
Image 2	0.5756	3.8260	1656.0166	11.2636	5.0356	6.0649
Image 3	0.8152	7.4259	2678.378	17.5957	7.3329	9.1378
Image 4	1.0747	12.9124	3543.2988	26.6574	10.0399	12.9377
Image 5	1.8757	33.5103	5641.1023	55.4430	21.1628	26.9086

Table 4: Precision, Recall and F-score values for Image 1 and Image 2 using Pixel-based classification. Values in boldface indicate the average value.

	Image 1			Image 2		
	Precision	Recall	F-score	Precision	Recall	F-score
Roads	0.889540	0.730460	0.802189	0.286645	0.32629	0.305186
Agr Z	0.912811	0.800453	0.852948	0.561734	0.576946	0.569239
Urb Z	0.628760	0.766913	0.690999	0.694606	0.845396	0.762618
Veg Z	0.658451	0.882978	0.754362	0.84423	0.843212	0.843721
Avg.	**0.772390**	**0.795201**	**0.775124**	**0.596804**	**0.647961**	**0.620191**

Table 5: Precision, Recall and F-score values for each image using Pixel-based classification. Values in boldface indicate the average value.

	Precision	Recall	F-score
Image 1	0.772390	0.795201	0.775124
Image 2	0.596804	0.647961	0.620191
Image 3	0.66972	0.715261	0.684503
Image 4	0.831632	0.780933	0.80491
Image 5	0.778437	0.741131	0.755813
Avg.	**0.729796**	**0.736097**	**0.728108**

Table 6: Precision, Recall and F-score values for Image 1 and Image 2 using Superpixel-based classification. Values in boldface indicate the average value.

	Image 1			Image 2		
	Precision	Recall	F-score	Precision	Recall	F-score
Roads	0.914880	0.737076	0.816409	0.223892	0.250646	0.236515
Agr Z	0.913373	0.815941	0.861912	0.605537	0.55185	0.577449
Urb Z	0.664871	0.838919	0.741823	0.673891	0.846333	0.750332
Veg Z	0.674830	0.881962	0.764616	0.845352	0.847048	0.846199
Avg.	**0.791988**	**0.818474**	**0.796190**	**0.587168**	**0.623969**	**0.602623**

http://www.imavs.org/

Table 7: Precision, Recall and F-score values for each image using Superpixel-based classification. Values in boldface indicate the average value.

	Precision	Recall	F-score
Image 1	0.791988	0.818474	0.796190
Image 2	0.587168	0.623969	0.602623
Image 3	0.674413	0.719813	0.688764
Image 4	0.834022	0.785795	0.808782
Image 5	0.775443	0.739259	0.753459
Avg.	**0.732606**	**0.737462**	**0.729963**

Table 8: Statistical results (B:best, W:worst, Av:average, Md: median and SD: standard deviation) for ten test for Image 1 and Image 2 using Superregion-based classification and evaluating half of the superpixels belonging to a superregion. Values in boldface represent average values.

	Image 1			Image 2		
	Precision	Recall	F-score	Precision	Recall	F-score
B	0.799210	0.821027	0.800211	0.594172	0.626659	0.603561
W	0.780585	0.799252	0.781955	0.558583	0.595353	0.575089
Av	0.789611	0.812634	0.791926	0.580797	0.614114	0.593997
Md	**0.79113**	**0.81579**	**0.79400**	**0.58189**	**0.616448**	**0.59545**
SD	0.006109	0.007718	0.006293	0.010209	0.011323	0.009934

Table 9: Statistical results (B:best, W:worst, Av:average, Md: median and SD: standard deviation) for ten runs for Image 1 and Image 2 using Superregion-based classification and evaluating a quarter of superpixels belonging to a superregion. Values in boldface indicate average values.

	Image 1			Image 2		
	Precision	Recall	F-score	Precision	Recall	F-score
B	0.777467	0.812371	0.782793	0.62784	0.71738	0.66644
W	0.734272	0.755387	0.724421	0.54104	0.58519	0.55893
Av	0.761531	0.791289	0.762080	0.57714	0.61632	0.59320
Md	**0.76166**	**0.79220**	**0.76482**	**0.57277**	**0.60958**	**0.58964**
SD	0.013380	0.016181	0.016911	0.02450	0.03704	0.02854

Table 10: Precision, Recall and F-score values for each image using Superregion-based classification. Values in boldface indicate the average value.

	Precision	Recall	F-score
Image 1	0.79113	0.81579	0.79400
Image 2	0.58189	0.616448	0.59545
Image 3	0.684791	0.753373	0.709578
Image 4	0.852012	0.777926	0.812681
Image 5	0.822926	0.788309	0.804562
Avg.	**0.746549**	**0.750369**	**0.743254**

superregion-based classification improved the misclassification into pink areas. With respect to regions labeled with the number 1 and 2 were improved. Finally those regions delimited by red rectangle preserves the behavior, due to any approach and our CNN model was not able to improve the misclassification. With respect to Image 2 (right column), the CNN model misclassified the roads, which were labeled as urban region. Furthermore, this problems happen in less degree with agricultural and vegetation zone. In these images, the plantations grow land share features with vegetation area, for this reason is classified as vegetation area.

4.6 Discussion

The accuracy performance of our CNN model approach can segment the most of image correctly, the average F-score for each approach classification is 0.732 approximately. This is better than [10], which is 55.52%, on the experiment

http://www.imavs.org/

Table 11: Precision, Recall and F-score values for each image using Superregion-based classification. Values in boldface indicate the average value.

	Precision	Recall	F-score
Image 1	0.76166	0.79220	0.76482
Image 2	0.57277	0.60958	0.58964
Image 3	0.664193	0.722956	0.679302
Image 4	0.845415	0.777604	0.809196
Image 5	0.812956	0.788685	0.798829
Avg.	**0.731398**	**0.738205**	**0.728357**

where RGB satellite images were employed. Furthermore, this value (0.732) is greater than the overall accuracy (60-70 percent) reported by [8]. The use of superpixels allow defining more precisely the borders between classes. With respect the use of superregions, this reduce the time required to segment an image, but the final semantic segmentation depend on the randomly chosen superpixels. Two factors can affect the performance of the CNN model, which are the height of the image and the shadows presented, due to several elements are misclassified.

5 CONCLUSIONS

The focus of this work lies on testing the performance of a CNN model trained with RGB aerial images, that is without additional information such as DMS or NDVI. Moreover, the altitude of the images is closer than satellite images, that is, images taken from 100-300 meters above the ground, which represent a challenging task because these images contain more details than satellite images. Due to its high pixel resolution, the computing effort to process an image is reduced, in the number of elements to be classified, by using superregions or superpixels. According to the results, our model can predict most of the pixels correctly. Note that our pixel-based approach produces the salt-and-pepper effect, which is decreased with the use of superpixels and superregions. Although the accuracy is similar in the three approaches presented in this work, the use of superpixels and superregions reduce significantly the required time to segment an image.

In future work, we aim at optimising our proposed methods to enable on-line processing. Our goal is to carry out the classification during operation flight, in particular, by using small drones.

ACKNOWLEDGEMENT(S)

Thanks to Magda-Nayeli González-Sánchez for her support to build the dataset of this work. And, all authors acknowledge support from INEGI and CONACyT through project No. 268528.

REFERENCES

[1] Yaning Yi, Zhijie Zhang, Wanchang Zhang, Chuanrong Zhang, Weidong Li, and Tian Zhao. Semantic segmentation of urban buildings from vhr remote sensing imagery using a deep convolutional neural network. *Remote Sensing*, 11(15):1774, Jul 2019.

[2] Y. Babykalpana and K. ThanushKodi. Classification of lulc change detection using remotely sensed data for coimbatore city, tamilnadu, india. *Journal of Computing*, 2(5):46–56, 05 2010.

[3] Robail Yasrab. Ecru: An encoder-decoder based convolution neural network (cnn) for road-scene understanding. *Journal of Imaging*, 4(10):116, Oct 2018.

[4] Matija Radovic, Offei Adarkwa, and Qiaosong Wang. Object recognition in aerial images using convolutional neural networks. *Journal of Imaging*, 3(2):21, Jun 2017.

[5] Guangming Wu, Xiaowei Shao, Zhiling Guo, Qi Chen, Wei Yuan, Xiaodan Shi, Yongwei Xu, and Ryosuke Shibasaki. Automatic building segmentation of aerial imagery using multi-constraint fully convolutional networks. *Remote Sensing*, 10(3):407, Mar 2018.

[6] Pascal Kaiser, Jan Wegner, Aurelien Lucchi, Martin Jaggi, Thomas Hofmann, and Konrad Schindler. Learning aerial image segmentation from online maps. *IEEE Transactions on Geoscience and Remote Sensing*, 55(11):6054–6068, 07 2017.

[7] Maciej Dzieszko, Piotr Dzieszko, Slawomir Królewicz, and Jercy Cierniewskski. Digital aerial images land cover classification based on vegetation indices. *Quaestiones Geographicae*, 31(3):5–23, October 2012.

[8] Ming-Jer Huang, Shiahn-wern Shyue, Liang-Hwei Lee, and Chih-Chung Kao. A knowledge-based approach to urban feature classification using aerial imagery with lidar data. *Photogrammetric Engineering & Remote Sensing*, 74:1473–1485, 12 2008.

[9] Martin Längkvist, Andrey Kiselev, Marjan Alirezaie, and Amy Loutfi. Classification and segmentation of satellite orthoimagery using convolutional neural networks. *Remote Sensing*, 8(4):329, Apr 2016.

[10] Sakrapee Paisitkriangkrai, Jamie Sherrah, Pranam Janney, and Anton van den Hengel. Semantic labeling of

http://www.imavs.org/

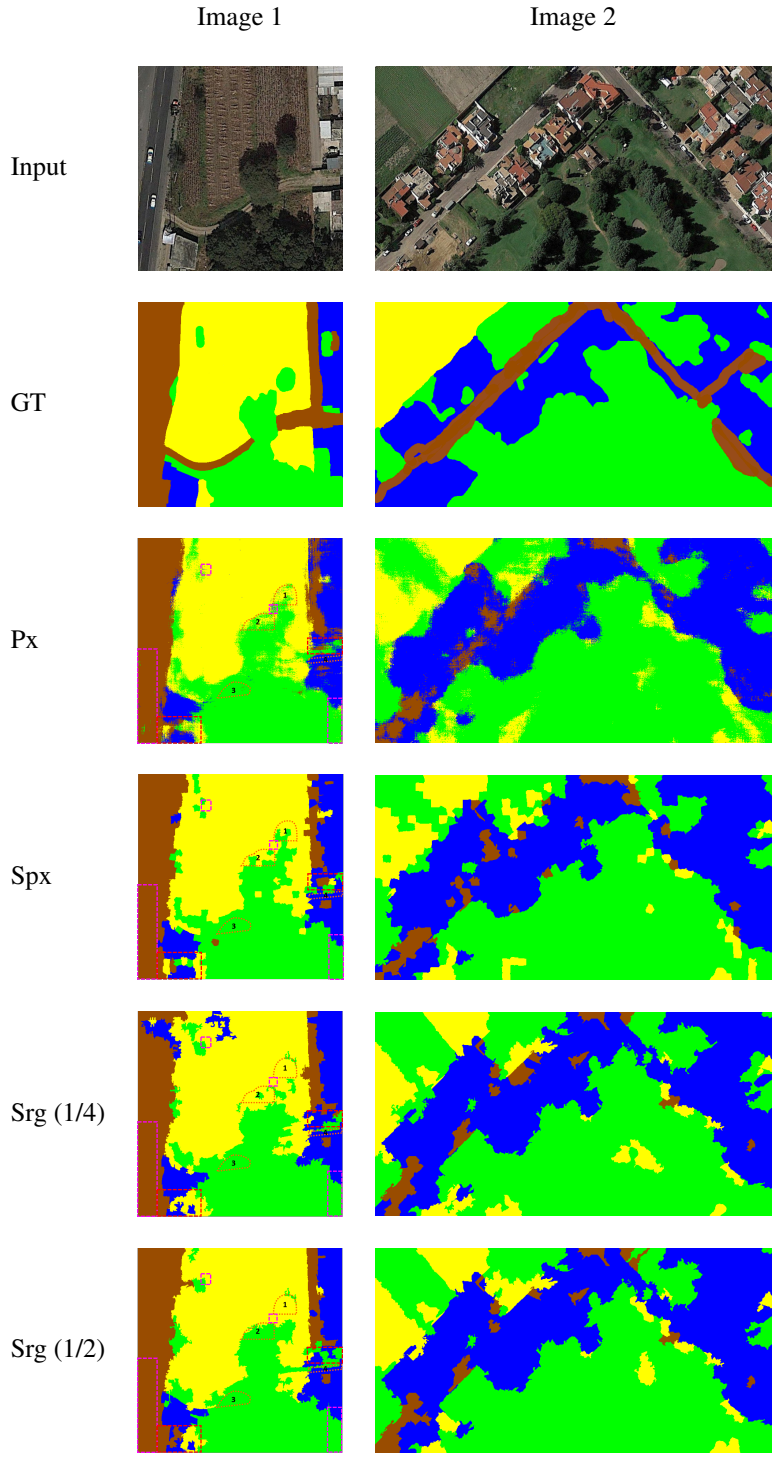


Figure 4: Output for each experiment: Px - pixel-based classification, Spx - superpixel-based classification, Srg 1/4- superregion-based classification 1/4, Srg 1/2 - superregion-based classification 1/2.

aerial and satellite imagery. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(7):2868–2881, 2016.

[11] Nicolas Audebert, Bertrand Le Saux, and Sébastien Lefèvre. Semantic segmentation of earth observation data using multimodal and multi-scale deep networks. In *Asian Conference on Computer Vision (ACCV16)*, pages 180–196, Taipei, Taiwan, 2016.

http://www.imavs.org/

[12] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[13] Chun Yang, Franz Rottensteiner, and Christian Heipke. Classification of land cover and land use based on convolutional neural networks. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-3:251–258, 04 2018.

[14] Lei Ding, Hao Tang, and Lorenzo Bruzzone. Lanet: Local attention embedding to improve the semantic segmentation of remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, pages 1–10, 2020.

[15] Jiankun Li, Wenrui Ding, Hongguang Li, and Chunlei Liu. Semantic segmentation for high-resolution aerial imagery using multi-skip network and markov random fields. In *2017 IEEE International Conference on Unmanned Systems (ICUS)*, pages 12–17, 2017.

[16] Dimitris Marmanis, Jermaine Wegner, Silvano Galliani, Konrad Schindler, Mihai Datcu, and Uwe Stilla. Semantic segmentation of aerial images with an ensamble of cnns. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, III-3:473–480, 06 2016.

[17] Kai Yue, Lei Yang, Ruirui Li, Wei Hu, Fan Zhang, and Wei Li. Treesegnet: Adaptive tree cnns for subdecimeter aerial image segmentation. *Computer Vision and Pattern Recognition*, 2018.

[18] Vikas Gurumurthy. Encoder-decoder based cnn and fully connected crfs for remote sensed image segmentation. 10 2019.

[19] Jamie Sherrah. Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery. *CoRR*, abs/1606.02585, 2016.

[20] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, page 1–14, 2015.

[22] David Stutz, Alexander Hermans, and Bastian Leibe. Superpixels: An evaluation of the state-of-the-art. *Computer Vision and Image Understanding*, 166:1–27, Jan 2018.

[23] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurélien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels. *Technical report, EPFL*, 06 2010.

[24] Imanol Luengo, Mark Basham, and Andrew French. Smurfs: Superpixels from multi-scale refinement of super-regions. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 4.1–4.12. BMVA Press, September 2016.

http://www.imavs.org/

Multi-Model Continual Learning for Camera Localisation from Aerial Images

Aldrich A. Cabrera-Ponce, Manuel Martin-Ortiz, and Jose Martinez-Carranza*

*Instituto Nacional de Astrofisica, Optica y Electronica (INAOE), Puebla, Mexico
Benemerita Universidad Autonoma de Puebla (BUAP), Puebla, Mexico

http://www.imavs.org/

Abstract

Camera localisation is the problem of estimating the camera pose from one or more images. Deep Learning (DL) has become one of the most popular methods with convolutional neural networks (CNN) to train models and thus obtain the camera pose in a scenario. However, such methods assume a scenario where the user must wait for the model to become available once ready after training, which has to be trained off-line and involving high computational processing time. This work presents a multi-model continuous learning approach for camera localisation that uses multiple models learned incrementally over time. Each model is learned for a certain segment of the camera trajectory, and a search engine based on histogram colour and keyframes to identify the corresponding learned model w.r.t. the current camera view. The continual learning process is a rehearsal strategy that repeats the old data with the new ones in a latent replay layer. On average, our performance achieved a processing speed of 150 fps with an accuracy of 0.71 to 0.85 for the camera pose estimation. We demonstrate the effectiveness of our system in a geo-localisation application where a model learns to associate aerial images to GPS coordinates continuously using our proposed strategy. The main goal is to provide multiple on-line learned models during the same flight as an alternative geo-localisation method that could prove helpful in cases of GPS failure.

1 Introduction

Camera relocalisation is the problem of obtaining the 6 degrees of freedom (DoF) concerning one or multiple images captured from a scenario. Visual localisation methods have been carried out to solve this problem using the images from a monocular camera. For example, some works acquire the camera's poses using feature matching [1], nearest neighbours [2], visual odometry (VO) [3], and simultaneous localisation and mapping (SLAM) systems [4]. The performance of these techniques has been adequate to acquire information that can solve the visual localisation problem. Nevertheless, with the recent increase in deep learning, works have addressed this problem to directly regress the camera pose from an image with a training model [5].

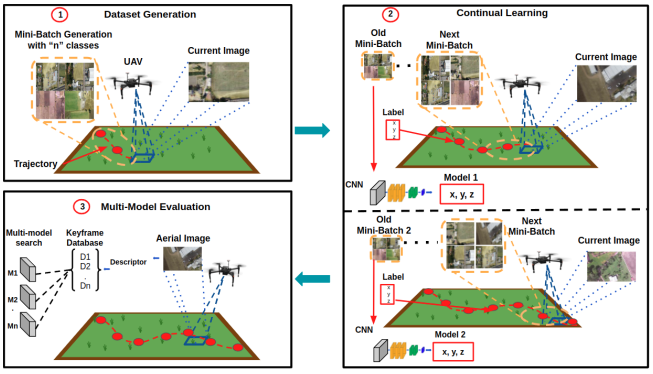


Figure 1: Multi-model continual learning for camera localisation. The framework consists of 3 steps: 1) Mini-batches generation; 2) On-line training using the mini-batches while new ones are created; 3) Frame search based on histogram colour and keyframes to identify the corresponding model. A video illustrating this process can be found at video link.

To carry out the aforementioned, models are trained from a dataset with labels of the camera positions to regress the 6-DOF camera pose from a single RGB image [6]. Nonetheless, traditional training has the limitation of not having a learning model ready for the same data collection mission. To mitigate this, it has been established to speed up the training time and pose estimation, reducing the architecture of a network [7]. On the other hand, continual training aims to learn the information on the fly, reducing the catastrophic forgetting in the training process [8].

With the above mentioned, a compact network and a method that reduces forgetting have achieved relevant results under scenarios requiring a fast training process. As a result, continual learning methods reduce the loss in the training process by keeping the information to a minimum during the same collection data mission, getting results simultaneous. Thus, a model is updated with current data without forgetting previous ones during training. Therefore, continual

*Email address(es): aldrich.cabrera@alumno.buap.mx

learning has a high impact in the area of robotics used in mobile applications with limited storage capacity [9].

Motivated by the above, we consider using a network based on a continual learning strategy with a rehearsal method, allowing the creation of multiple models with information about the poses in a specific area. We adopt the work presented in [10] to repeat the previous data in a latent layer and merge it with the current information. Besides, based on localisation modules in SLAM systems, an image is localised using feature matching between its descriptors and a bag of words (BoW). Thus, we propose a frame searcher to determine the keyframe corresponding to the camera view and identify the learned model w.r.t. the current view (Figure 1). Finally, we argue that using multi-models with information from the camera trajectory segments can give us a result close to the value of the Ground Truth, allowing re-localise a camera.

In order to present this work, our paper is organised as follows. Section 2 provides a literature study of the continual learning approaches for classification and localisation tasks. Section 3 describes the process of generating a multi-model system using the AR1* continual learning method and the generation of mini-batches. The experimental design and results obtained compared with ORB-SLAM2 are conveyed in Section 4. Finally, conclusions and future work are outlined in Section 5.

2 RELATED WORKS

Image-based localisation has become an efficient method in several robotics tasks, obtaining a camera's pose using computer vision methods. For example, a UAV's camera can capture images to determine the position in a scenario, thus achieving its localisation. ORB descriptors are a technique used for camera localisation by building a map of the scenario [4]. On one side, deep learning have been used to get the camera's pose using regression in combination with 3D structures [11], visual odometry [12] and SLAM systems [13, 14] in monocular, stereo, and RGB-D cameras. These works employ a preprocessing that aligns in-depth features with a reference 3D model, crops sub-images, and uses inverse depth to learn better the image's shape, appearance and feature.

However, the works aforementioned require 3D models, crop images and the use of depth to achieve an accurate result. On the other hand, PoseNet [5] is a CNN designed to estimate the camera's pose from a single image without using depth and 3D models. Therefore, several works have used PoseNet to improve and change its architecture to get a fast and accurate pose estimation. For example, in [7] shows a brief analysis of these architectures, proposing a compact network to accelerate the learning process and pose estimation. Nevertheless, PoseNet is an end-to-end architecture that involves collecting a large dataset. Thus, new works have focused on continuous training within the robotics [9] to learn new data without forgetting the previous ones in detection and classification tasks [15, 16].

The concept of continual learning has led to creating datasets from a sequence of partial experiences where all data is unavailable at once. For example, in [17] developed CoRE50, a dataset for continual learning in object recognition tasks, and [18] presents CLRS dataset for scene classification, partitioning the training batch. However, the catastrophic forgetting is still unsatisfactory in results, arguing the design of a method to restrict this loss. Unlike these works, an elastic network with a multistable behaviour can allow the development of different configurations to learn continuously without forgetting the previous states, avoiding thus the knowledge forgetting [19].

Another work presents a continual SLAM [20] proposing a dual-network based on a VO model to produce online odometry estimates and generalise the long-term learning, using consecutive frames, image warping, and dense 3D map. However, since this method achieves the localisation of a camera by merging 3D depth maps, it may fall into an estimation error if the depth is not acquired correctly. To avoid this, [21] analyses and identifies previously unobserved phenomena of gradient-based optimisation using visual data and an adaptable buffer for the geo-localisation of an image. Besides, they keep the images with lower noisy time stamps in the form of the default date and time for training.

Finally, rehearsal methods based on repeating the previous data for visual localisation tasks achieve promising results. For example, using an adaptive buffer keeps samples in subsets to update the training model over current information and previous ones [22]. In [10], presents the AR1* method with a latent replay layer to retain volumes of information of the previous data as new information is created in mini-batches. The latter has been carried out in a localisation methodology using multicamera learning and updating the model while acquiring new information [23]. Motivated by these works, we implemented a multi-model approach based on a continual learning method called AR1*, using a latent replay layer for camera localisation. In addition, we built a frame search based on histogram colour and keyframes to identify the corresponding model.

3 METHODOLOGY

This section presents the methodology carried out to develop this work. We carry out our multi-model continual learning framework in 3 steps. Firstly, we generated the dataset with aerial images and GPS coordinates into mini-batches for the training. Afterwards, we use a colour histogram like an image descriptor to determine the corresponding keyframe. Thus, it will indicate which model corresponds with the image to perform the evaluation. Finally, we present the continual training with the latent replay strategy.

3.1 Mini-Batches generation

For the dataset generation, we collected aerial images with GPS information through the Robotic Operating System

http://www.imavs.org/

http://www.imavs.org/

(ROS) and our Ground Control Station (GCS), acquiring the frames from the video stream of the UAV. Then, we manually carried out 4 flight missions to obtain the dataset, splitting it into small sets of images with dimensions of 128×128 and converting the GPS to metres. Then, we generate the labels when we create a new class with a number of pictures defined with its labels in terms of indexes and create the mini-batches every two classes. Furthermore, since the network is an architecture based on a classification model, we define a representative mean flight coordinate for each class, merging the individual coordinates into one. This way, we store the mean flight coordinates in a file to represent the classes and UAV trajectory.

To better understand the mean flight coordinates stored in the file, we show the coordinates of one of the trajectories in Table 1, where the index represents the class belonging to a part of the path. Thus when the model gives us a result in terms of index, we look for in the text file to regress the mean flight coordinate and geolocalise the camera. Also, we show in Figure 2 the trajectories made in the flight missions with the UAV where we collected the data.

Table 1: Distribution of the classes associated with indexes representing the labels with discrete flight coordinates.

Index	Mean flight coordinates
0	x: -0.28260, y: 15.8941, z: 100.81
1	x: -14.3834, y: 50.8135, z: 100.94
2	x: -233.215, y: 228.076, z: 100.71
3	x: -351.116, y: 288.614, z: 100.34
4	x: -248.701, y: 341.021, z: 100.38
5	x: -125.802, y: 252.147, z: 100.52
6	x: 60.32710, y: 145.127, z: 100.50

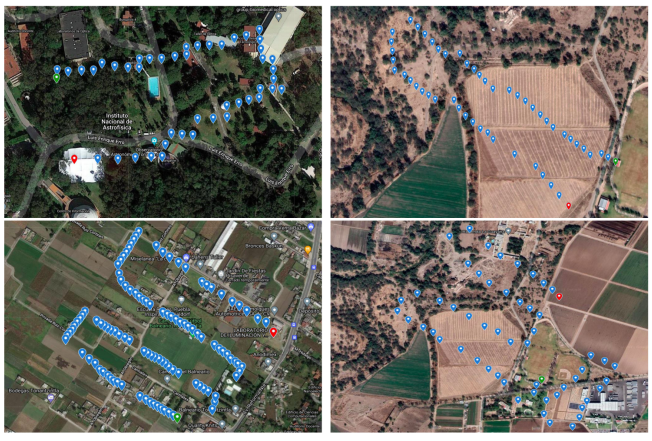


Figure 2: Manually performed flight trajectories. Each blue waypoint corresponds to the GPS coordinates; the red waypoint represents the beginning and the green the end of the trajectory.

3.2 Histogram image descriptor

To create the searcher that allows us to choose the model for the evaluation correctly, we propose using a simple descriptor based on the colour histogram of the image. This histogram applies a colour distribution in the pictures, causing them to have similar distributions or equal even in those that are not. Therefore, we perform the following steps to generate a robust descriptor to deal with this problem. The steps are described in a list to understand the procedure performed:

1. We convert the image colour space from BGR to HSV (Hue, Saturation, Value) to obtain robustness in the histogram.
2. We define the number of bins in our histogram with a dimension of 8 bins for the Hue, 12 bins for saturation, and 3 bins for the value channel. Total dimension of the feature vector: $8 \times 12 \times 3 = 288$.
3. We calculate a mask from the centre of the image and divide it into quadrants. Thus, we calculate the histogram in 5 regions of the image instead of a global one.
4. We update the features extracted from the five quadrants with values from 0 to 255.
5. Finally, the images are represented and quantified using a list of 288 floating-point numbers.

In this way, we calculate the colour histogram from a divided image into five different regions: 1) the top-left corner; 2) top-right; 3) bottom-right; 4) bottom-left; 5) and the centre of the image. We show this example in Figure 3. Once the histogram is obtained, we calculate the features for each keyframe that we will use to find the model. Thus, we extract the floating points from the histogram and save them in a list with their respective features. In addition, we have a feature vector from 5 regions with a dimensionality of 5 x 288 = 1440, where each image will be quantified and represented using 1440 numbers.

Finally, we compare the frame with the keyframe to find the match between their features. That is to say, we extract the histogram information of the current frame and compare it with the keyframe descriptor using the chi2 distance. Thus, we get histogram information from the five regions on the images, ordering the results from lowest to highest, where a value close to zero represents the similarity with the keyframe. Then, we load the model corresponding to this keyframe to start the evaluation. Figure 4 presents a diagram of this process to find the model through a distance metric, taking two feature vectors as inputs. It is worth mentioning that we create the keyframes just when we generate the mini-batches, so if the value of the histograms does not match, it will return a model that does not correspond to the current camera view.



Figure 3: Image divided in 5 regions.

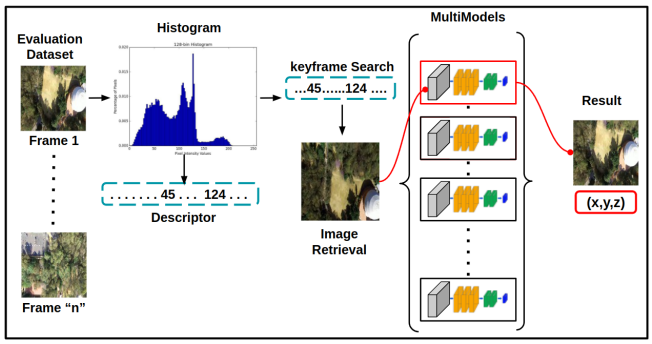


Figure 4: Overview to search for the keyframe with an evaluation frame using the descriptor vector to identify the corresponding model.

3.3 *Continual Training*

For multi-model continual training, we rely on the architecture presented in the work [10], showing a method of latent replay in the hidden layer *pooling6*. This process consists of training a dataset from mini-batches whose classes are spread in them, storing a portion of activation volumes belonging to each class. In the training process, the method slows down learning in the lower layers and leaves the upper layers free. That is to say, it slows down the learning at the layers below the pooling6 layer, saving a small fraction of patterns for each mini-batch. Thus, when new information arrives (a new mini-batch), the memory with previous data is merged and updated with the current data. In this way, the patterns of both mini-batches are kept constant by repeating the data in the latent layer. Finally, we perform this process until the last mini-batch generated will train.

This method, called Auto-Regressive Model (AR1*), is based on a rehearsal strategy which consists of the continuous repetition of previous data to avoid catastrophic forgetting of the learning. Each mini-batch represents new information generated from the dataset, training them one by one. We thus update a learning model when new data arrives, generating a single model with all the information learned. However, to develop our multi-model approach, we create a model for each mini-batch learned, arguing that each has information more accurate to the current data than the previous one. Finally, we have several models corresponding to the mini-batches for evaluation tests.

To carry out the continual training, we take the following parameters: 1) SGD Optimiser; 2) Batch size: 128; 3) Epochs: 4; 4) Cross-Entropy as loss function; 5) Learning Rate: 0.001; 6) Pool6 as latent layer; 7) 1500 patterns to storage in the external memory. These patterns are the information acquired from the previous and current data to update the learning model.

4 Experiments and Results

Hereafter, we present the experiments to compare the accuracy and processing speed results using a learning model and our multi-model approach. We should note that the search engine for a keyframe using a descriptor based on the colour histogram of the image is essential to establish the chosen model. Given this scenario, we perform the experiments on Ubuntu 20.04, Python 3.6, PyTorch 1.4.0 for the network training and evaluation, OpenCV 3 and ROS Noetic on a computer with an Nvidia Geforce 960 M with 8 GB of RAM.

4.1 *Histogram Results*

The first experiment compares the search result using three bins to generate the histogram. Thus, we calculate the histogram of each evaluation dataset image of trajectory one and search using the chi2 distance for the corresponding similarity to obtain the model. Table 2 shows the number of times the search acquired the correct model and a success rate indicating how effective the search is with the defined bins.

Table 2: Results of the searching model using a descriptor vector.

Bins	Correct	Success Rate
(10, 5, 3)	78	54.16 %
(12, 16, 3)	111	77.08 %
(8, 12, 3)	125	**86.90 %**

This result indicates that the third set bins achieve a better result w.r.t colour histogram, finding the correct keyframe using the descriptor to identify the corresponding model. However, we argue that a descriptor based on SIFT and ORB could improve this search result using visual and binary features corresponding to the image.

4.2 *Evaluation with a Model*

In the second experiment, we used a single learning model without the descriptor searching that determines the keyframe, training and updating the weights in the model

http://www.imavs.org/

when new mini-batches arrive in the network. Thus, the previous and current classes' patterns are replayed in the latent layer, updating the features and generalising the knowledge for each mini-batch trained. To illustrate the accuracy results in this experiment, we evaluate the CNN with the evaluation data of the 4 flight paths, obtaining an index corresponding to the class label with its mean flight position. Then, in Table 3, we present the results obtained, showing the number of frames present in the evaluation dataset, the number of correctly classified frames, the accuracy, and the processing speed per image expressed in fps.

Table 3: Accuracy results with the evaluation dataset using a single model.

Trajectory	Frames	Correct	Accuracy	Fps
1	144	89	0.6180	152.397
2	117	73	0.6239	153.044
3	84	70	0.8333	154.096
4	606	378	0.6237	151.564

We note that training presents an accuracy of around 0.62 in most of the paths whose traversed trajectory lengths are: 1.0 km for the first, 3.0 km in the second, 4.6 km for the third, and 6.4 km for the last. The best result was in the third flight achieving an accuracy of 0.83 with a high amount of frames evaluated correctly. Hence, we argue that the algorithm is limited in storing the volumes of information, causing catastrophic forgetting to occur the more classes it learns, even using a latent layer. Consequently, we expose that a multi-model approach for the camera localisation can have a better result using exclusively on the model corresponding to the camera view class.

4.3 Evaluation with a Multi-Model Approach

In the last experiment, we evaluated our multi-model approach with the keyframe searching to identify the corresponding model to the class and camera view. We used the bins of (8,12,3) analysed in the first experiment to calculate the histogram and create the feature vector descriptor. Afterwards, we compare our approach with the ORB-SLAM2 re-localisation module, presenting the accuracy and processing speed results. To compare us with ORB-SLAM2, we created a map with the training dataset and localised the evaluation images with the ORB-SLAM2 re-localisation module to find the coordinates w.r.t the Ground-Truth. Thus, we take the distance between frame and keyframe to determine whether the coordinate corresponds to a close keyframe. Finally, in Table 4, we present the results obtained from the four trajectories and the comparison with ORB-SLAM2, and in Table 5, we show the processing speed results.

These results show that ORB-SLAM2 presents a low localise on trajectory 1, indicating that the scenario is complex to obtain a localisation close to the training dataset. Furthermore, the ORB-SLAM2 re-localisation module does not find ORB descriptors that match the evaluation images, arguing that the scenario presents elements that confuse the system. Likewise, path 4 has an accuracy just above 0.5, meaning that half of the path cannot correctly locate the camera. Nonetheless, our multi-model system finds some coordinates of the evaluation images, localising thus the camera in these areas.

Table 4: Comparison accuracy results with the evaluation dataset using a multi-model and ORB-SLAM2.

Approach	Traj. 1	Traj. 2	Traj. 3	Traj. 4
ORB-SLAM2	0.034	0.913	0.869	0.584
Multi-Model	0.735	0.817	0.856	0.714

Table 5: Comparison processing speed results with our multi-model approach and ORB-SLAM2 (Fps).

Approach	Traj. 1	Traj. 2	Traj. 3	Traj. 4
ORB-SLAM2	85.47	83.33	92.59	89.28
Multi-Model	150.19	151.65	153.27	153.55

For the above, we argue that a multi-model approach using continual learning based on latent replay achieves a better result by identifying the corresponding model with the camera view. Besides, searching keyframes can recover the model created in that trajectory segment using only the descriptor vector without geometry calculations or depth images. Eventually, the processing speed tests show a superiority in terms of time to run the algorithm, which we can use during flight missions with a UAV in external scenarios.

To better understand the results obtained in the earlier experiments, we present in Table 6 the RMSE metric to observe how far the obtained values are from the ground truth. Thus, a minimum value represents a better fit to the accuracy measure with which the algorithm performs the evaluations. Note that our approach presents significant results in contrast to using a single model for camera localisation tasks. Finally, it should be noted that if the UAV enters an unknown area or outside the training area, the algorithm gives erroneous data. To deal with this, we plan to use a flag where we determine that the UAV has left the zone and thus extend the training with the following area data.

Table 6: Results of the RMSE metric using the 3 approaches performed in the experiments: with a single model, ORB-SLAM2, and our multi-model approach.

Approach	Traj. 1	Traj. 2	Traj. 3	Traj. 4
ORB-SLAM2	5.9389	0.4090	1.8563	8.2484
Single Model	2.1730	2.3056	4.4507	7.8642
Multi-Model	1.8671	1.6641	2.7080	7.0858

http://www.imavs.org/

5 CONCLUSIONS

We have presented a continual learning approach to localise a monocular camera using multi-models and a keyframe search engine to identify the representative model given the current camera view. Our goal has been to use the histogram colour of the images in 5 regions to search a close keyframe and choose the learned model, obtaining thus a label with a mean flight coordinate for camera localisation. Furthermore, it helps to extract and match similar features between the current camera view and keyframe, giving access to the corresponding model to evaluate the image instead of using a global model. In addition, the advantage of continual learning in a latent replay approach allows for learning data in small sets, maintaining previous information while updating the model with new data.

The proposed algorithm exhibits a suitable accuracy for camera localisation in combination with an image searching strategy, being efficient in small, medium and large trajectories. Furthermore, we have also demonstrated the applicability of a rehearsal strategy to improve the localisation using our scheme compared with SLAM systems. Overall, we achieve an accuracy result of 0.73 for short trajectories and 0.71 for long ones while maintaining a fast processing speed. Nonetheless, the feature descriptor is a topic to discuss because if the colour histogram returns a different keyframe, the model will give us another result and thus a wrong position.

Finally, we have observed that tests performed with continual learning based on mini-batches can be used to train new information in unseen areas while maintaining knowledge of the previous scenario. To this end, we propose extending the experiments to different environments to test the algorithm's effectiveness when the UAV visit new zones. In addition, we plan to use the method with a regression network to estimate the full 6 DoF camera position for future work.

http://www.imavs.org/

ACKNOWLEDGEMENTS

The first author is thankful for his scholarship funded by Consejo Nacional de Ciencia y Tecnología (CONACYT) under the grant 802791.

REFERENCES

[1] Peter Hansen, Peter Corke, and Wageeh Boles. Wide-angle visual feature matching for outdoor localization. *The International Journal of Robotics Research*, 29(2-3):267–297, 2010.

[2] David Wong, Daisuke Deguchi, Ichiro Ide, and Hiroshi Murase. Single camera vehicle localization using feature scale tracklets. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 100(2):702–713, 2017.

[3] Ramon Gonzalez, Francisco Rodriguez, Jose Luis Guzman, Cedric Pradalier, and Roland Siegwart. Combined visual odometry and visual compass for off-road mobile robots localization. *Robotica*, 30(6):865–878, 2012.

[4] 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.

[5] 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.

[6] Alex Kendall and Roberto Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *2016 IEEE international conference on Robotics and Automation (ICRA)*, pages 4762–4769. IEEE, 2016.

[7] 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*, pages 1–11, 2022.

[8] Felix Wiewel and Bin Yang. Localizing catastrophic forgetting in neural networks. *arXiv preprint arXiv:1906.02568*, 2019.

[9] Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Dıaz-Rodrıguez. Continual learning for robotics. *arXiv preprint arXiv:1907.00182*, pages 1–34, 2019.

[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] Paul-Edouard Sarlin, Ajaykumar Unagar, Mans Larsson, Hugo Germain, Carl Toft, Viktor Larsson, Marc Pollefeys, Vincent Lepetit, Lars Hammarstrand, Fredrik Kahl, et al. Back to the feature: Learning robust camera localization from pixels to pose. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3247–3257, 2021.

[12] Felix Ott, Tobias Feigl, Christoffer Loffler, and Christopher Mutschler. Vipr: visual-odometry-aided pose regression for 6dof camera localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 42–43, 2020.

[13] Shilang Chen, Junjun Wu, Qinghua Lu, Yanran Wang, and Zeqin Lin. Cross-scene loop-closure detection with continual learning for visual simultaneous localization and mapping. *International Journal of Advanced Robotic Systems*, 18(5):17298814211050560, 2021.

[14] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Advances in Neural Information Processing Systems*, 34, 2021.

[15] Keval Doshi and Yasin Yilmaz. Continual learning for anomaly detection in surveillance videos. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 254–255, 2020.

[16] Kishan Parshotam and Mert Kilickaya. Continual learning of object instances. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 224–225, 2020.

[17] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *Conference on Robot Learning*, pages 17–26. PMLR, 2017.

[18] Haifeng Li, Hao Jiang, Xin Gu, Jian Peng, Wenbo Li, Liang Hong, and Chao Tao. Clrs: Continual learning benchmark for remote sensing image scene classification. *Sensors*, 20(4):1226, 2020.

[19] Menachem Stern, Matthew B Pinson, and Arvind Murugan. Continual learning of multiple memories in mechanical networks. *Physical Review X*, 10(3):031044, 2020.

[20] Niclas Vödisch, Daniele Cattaneo, Wolfram Burgard, and Abhinav Valada. Continual slam: Beyond lifelong simultaneous localization and mapping through continual learning. *arXiv preprint arXiv:2203.01578*, 2022.

[21] Zhipeng Cai, Ozan Sener, and Vladlen Koltun. Online continual learning with natural distribution shifts: An empirical study with visual data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8281–8290, 2021.

[22] 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.

[23] Aldrich A Cabrera-Ponce, Manuel Martin-Ortiz, and J Martinez-Carranza. Continual learning for multi-camera relocalisation. In *Mexican International Conference on Artificial Intelligence*, pages 289–302. Springer, 2021.

http://www.imavs.org/

Leveraging a Neural Pilot via Automatic Gain Tuning using Gate Detection for Autonomous Drone Racing

L. Oyuki Rojas-Perez, J. Martinez-Carranza*
Instituto Nacional de Astrofísica, Óptica y Electrónica Puebla, Mexico

http://www.imavs.org/

Abstract

Autonomous Drone Racing (ADR) has pushed the development of artificial pilots that enable a drone to fly autonomously on a race track. DeepPilot is an artificial pilot trained with a Deep Learning (DL) methodology for ADR. Given a set of camera images, this neural pilot estimates flight signals to navigate the drone throughout the race track. However, DeepPilot requires noise filters in output values to smooth out peaks in the flight signals, which produce oscillatory behaviour and jolts. Seeking to improve the performance of the artificial pilot, we propose to use a gate detector, also based on DL, to detect gates and use the position of the gate on the image in a state machine that automatically calculates the corresponding gains. To assess our approach, we carried out experiments in a simulation environment with a racetrack composed of 18 gates at different heights and orientations. We present a comparison with manually-set gains versus our automatic gain tuning approach. With the latter, the drone completes more than 15 consecutive laps on the same racetrack without collisions. Our approach was also successful in the scenario where some gates change position dynamically.

1 Introduction

Recent solutions to Autonomous Drone Racing (ADR) have proposed to use a deliberative approach involving path planning, control and trajectory tracking via Model Predictive Control and Reinforcement Learning to find optimal trajectories and control flight signals to navigate a race track as fast as possible [1, 2, 3]. Some of these approaches have been successful in demonstrating performance comparable to a human pilot [4]. However, this strategy relies on high-frequency and accurate pose estimation of the drone and knowledge of the position of the gates in the race track.

In contrast, other approaches have explored a more reactive strategy where no drone position or race track knowledge is available to the artificial pilot. Instead, the latter is *trained* to regress instantaneous flight signals from an image captured with a camera on board the drone [5, 6]. Deep Learning has become the viable option for implementing neural models that learn flight signals from camera images for ADR. For example, in our previous work, DeepPilot [6], we present an artificial pilot based on a Convolutional Neural Network (CNN). That includes temporal information for flight signal estimation by using six images equally sampled in one second (i.e., every five images per second). These images form a single image mosaic associated with their corresponding flight signals.

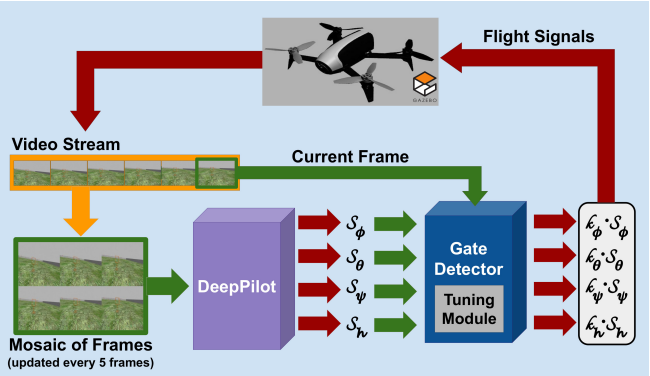


Figure 1: Schematic view of our approach where we propose to use a gate detector to leverage the performance of DeepPilot [6], a neural pilot trained to estimate flight signals $(S_\phi, S_\theta, S_\psi, S_h)$ from camera images. We propose to use the gate position detected on the current image to automatically tune a set of gains $(K_\phi, K_\theta, K_\psi, K_h)$ to replace the noise filter used by DeepPilot to avoid oscillatory behaviour and signal saturation. A video illustrating our approach can be seen at `https://youtu.be/49OzJCHTJu4`

DeepPilot has been shown to enable a drone to navigate on a race track with gates randomly placed and at different orientations and heights. However, the estimated flight signals should be noise filters in output values to smooth out peaks in the flight signals, which produce oscillatory behaviour and jolts. Another disadvantage is that if we change the platform, for example, the Bebop 2 simulated in RotorS [7], DeepPilot provides the incorrect proportion of the signal value. It means that the estimated flight signals are correct regarding what flight signal has to be sent to the drone (angle in roll, angle in pitch, velocity in yaw, velocity in altitude)

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

but with an incorrect proportion of the signal value. There are two options to solve this: 1) tuning a set of constants that act as *gains* to improve the performance of the artificial pilot and 2) providing more examples to the training set. However, this would involve the generation of large data sets and long training times.

Motivated by the fact that gain tuning would be preferable over large data generation, we propose using a gate detector's output to tune these gains automatically. DL has also been effectively used to solve the problem of gate detection on an image in the context of ADR [8], including its use to implement a flight controller [9]. However, given that the gate detector does not provide 3D information about the gate, we can not use the controller to control the yaw signal when the gate is observed in a skew view. Nevertheless, DeepPilot has learned to estimate a yaw signal under this scenario. Thus, we propose to use the gate detector, implemented with an SSD-based model [10] to leverage the performance of DeepPilot via automatic gain tuning. Figure 1 illustrates our full approach: first, flight signals are obtained with DeepPilot, which receives as input a six-image mosaic. Next, the gain tuner receives DeepPilot's output and the output of the gate detector, which uses the current image to detect the gate, whose position will calculate the value of the corresponding gain for each flight signal. Finally, we will send the product of the estimated flight signals multiplied by the gains to the drone.

We performed several experiments in simulation comparing the performance of DeepPilot without gain multiplication, DeepPilot with gains set manually, and DeepPilot with gains tuned using our proposed approach. Our results indicate a significant improvement over the former options. We also include an experiment where the drone performs several flights on the same track, and in some laps, the gates are changed in position to assess whether the gains adjust adequately, which effectively happened as expected.

For a more detailed explanation of our approach, this paper has been organised as follows: Section 2 summarises relevant related works; Section 3 presents our proposed methodology; our experimental framework is described in Section 4; conclusions and future work are discussed in Section 5.

2 RELATED WORK

Most ADR techniques are based on sensing, perception tasks, control, path flight, and localisation relative to the gate or global localisation using external sensors [1]. In most strategies, cameras were the principal sensor used to interpret the environment, gate or obstacle detection and localisation of the vehicle on the race track.

However, the latency from one frame to the next affects the vehicle's reaction to the race track and its speed. This latency and the blind spots (when the drone crosses the gate) are why the authors use complementary information obtained from IMU [9, 11, 12, 13, 14], LiDAR sensors [15, 8, 1], ultrasonic sensors and optical flow [1].

Concerning perception tasks, a popular open-source computer vision library, OpenCV, has been used to perform colour and contour detection. However, these techniques are susceptible to lighting conditions. Some authors [15, 8, 1, 10, 16, 17] have replaced computer vision techniques with deep learning, using open source libraries such as Caffe, Tensorflow and Keras.

Another important task for ADR is the flight path calculation. For this is necessary to use localisation systems such as SLAM, Visual Odometry, Semi-direct Visual Inertial Odometry (SVIO) or position estimation methods based on the IMU [1], using either the Kalman Filter or the Extended Kalman Filter. Even the trajectory planning can be modified by the gate location [15, 8, 1, 10], relative position [18, 13, 19, 9, 20], actions [1], velocity, or even the drone's direction [18, 16].

In contrast to visual localisation, authors in [4] present a solution using external sensors. In this, a drone controlled by artificial intelligence flew faster than three human pilots on a race track, reaching a maximum speed of 16 m/s. However, this approach requires the gates' global position to generate a time-optimal trajectory through the race track and execute the flight path inside a specific area where 36 VICON cameras are set. Therefore the flight path must be pre-calculated and is only useful for static race tracks.

The authors in [21] highlight that the most significant difference between human pilots and artificial pilots guided by a model for trajectory planning and control is that human pilots continuously explore and learn from experience and improve the visibility of gates by choosing an entry angle. In contrast, the based trajectory planning and control approaches do not consider experiences. Another important difference is that the artificial pilot relies on a position tracking system while human pilots perform a vision-based state estimation.

According to the works discussed in this section, we conclude that the solutions presented at ADR have difficulties interacting with dynamic environments and do not consider experience. Instead, most of the works assume that the track is static or use external sensors that allow them to optimise a set of waypoints corresponding to the global gate position. Motivated by this; we propose an approach based on visual information to leverage the behaviour of a neural pilot called DeepPilot and a Single Shot Detector for gate detection. For this approach, we don't require a model for trajectory planning, a global position nor a control methods to guide a drone on a racetrack.

3 METHODOLOGY

3.1 Flight Signal Estimation

DeepPilot is a computational model that associates a set of images with a flight signal. These signals represent the angular position of the drone body structure at the roll and pitch angles, thus producing a translational motion at those angles, the rotational speed at the yaw angle, and the verti-

http://www.imavs.org/

cal speed referenced to altitude. The values of these 4 flight signals $(S_\phi, S_\theta, S_\psi, S_h)$, estimated by the DeepPilot model, are passed to the drone's internal controller, thus allowing the drone to navigate autonomously through the race track gates, assuming that the next gate becomes visible immediately after crossing the current gate.

The DeepPilot architecture comprises three parallel branches to infer roll and pitch together but yaw and altitude separately. Each branch has four convolutional layers with three inception modules, one fully connected layer, and a regressor layer.

3.2 Gate Detector

We use the Single Shot Detector (SSD) network [22], a fast multi-category object detector, for gate detection. The SSD combines predictions on multiple feature maps of different sizes, producing detections at high and low levels of the image by applying convolution filters. The SSD network architecture is based on a VGG16 image classification network and an auxiliary structure composed of multiple convolution layers. The auxiliary structure obtains multi-scale feature maps and convolutional predictions to know the bounding box displacement and the box's position relative to the location of each feature map.

In this work, we use a reduced variant called SSD7; this is a small optimised network that uses an auxiliary structure composed of 7 convolution layers, which reduces the training and search time. The model used to detect the gate in a racetrack was trained with images obtained from the Gazebo simulator, and real environments, indoors and outdoors [9, 10].

3.3 Gain Tuning

Figure 1 shows the general diagram of our approach. First, the gazebo video stream from the Bebop2 platform camera is acquired to generate a mosaic composed of six frames updated every five frames, used as input to DeePliot. This mosaic allows DeepPilot to obtain temporal information from the environment to estimate the four flight signals $(S_\phi, S_\theta, S_\psi, S_h)$. It is important to mention that for this work, we will use the raw DeepPilot signals; we do not implement the noise filter presented in [6]. Next, the Gate Detector requested the four flight signals and the current frame of the video stream. In this block, the SSD7 network detects the gate on the image. Then based on the detections, the rules set in the gain tuning module are executed. If the Gate Detector does not provide detection, the following base gains $k_\phi = 1$, $k_\theta = 0.5$, $k_\psi = 5$ and $k_h = 1.2$ are used, obtained by empirical experimentation. Finally, the gains are multiplied by the corresponding flight signals estimated by DeepPilot to send to the Bebop2.

Our aim is to perform automatic gain tuning using the SSD7 gate detector. To this end, we identify four cases during the flight on the racetrack: 1) slow translation at the front, 2) errors on the edge of the image that corresponds to roll values, 3) roll oscillations at the front of the gate and 4) slow

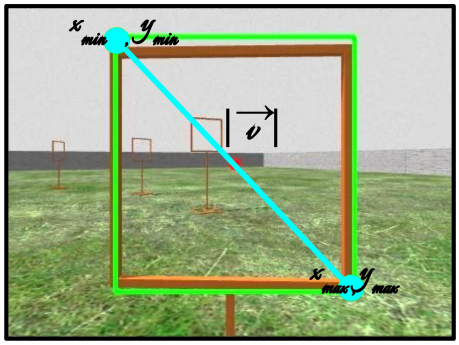


Figure 2: Information provided from the gate detector, where $x_{min}, y_{min}, x_{max}$ and y_{max} are used to select the closest next gate by calculating the largest vector norm $|\overrightarrow{v}|$.

rotation when the gates are skewed. To solve deal with each one of these cases, we design rules for k_ϕ, k_θ and k_ψ based on the gate detection. First, we select the next closest gate on a racetrack and for this, we calculate the vector norm $|\overrightarrow{v}|$ of the detections as shown in Equation 1, where $x = [x_{min} - y_{min}]$ and $y = [x_{max} - y_{max}]$ are obtained from the gate bounding box, then we select the largest vector norm, as shows Figure 2.

$$|\overrightarrow{v}| = \sqrt{x^2 + y^2} \tag{1}$$

We define the first case by using the gate detection area. For example, under the following rule, if the centroid of the gate detection is on the limit indicated in Figure 3, we obtain the value of k_θ using Equation 2, where Det_{area} is the area of the current detection and the $area_{max}$ has a value of 70 (set manually). With this condition, k_θ has a higher value when Det_{area} is smaller. Therefore the vehicle flies faster toward the gate and decreases k_θ value when the area increases. Another condition that affects the k_θ value is when the centroid of the gate detection is on the region indicated by Figure 3 and when S_ϕ and S_ψ provided by DeepPilot are equal to zero; in this case, k_θ has a value of two (twice the speed).

$$k_\theta = \frac{area_{max}}{Det_{area}} \tag{2}$$

The second case is that where the DeepPilot model does not correctly provide the roll signal when the gate is too close to the horizontal image edge, see Figure 4. Even when the gate is inside the viewing area, DeepPilot indicates $S_\phi = 0$. For this reason, we designed the following rule, when the centroid of the gate is at the edges of the image and the S_ϕ is equal to zero, the S_ϕ signal is replaced by 1 or -1 depending on the which edge the gate was spotted. In this way, the drone is forced to respond in the direction of the gate. With this signal, the gate is again inside the viewing area, the zone where DeepPilot provides S_ϕ with a non-zero value.

The third case is when oscillations of S_ϕ provoke the following issues: the vehicle stays in front of the gate for a long

http://www.imavs.org/

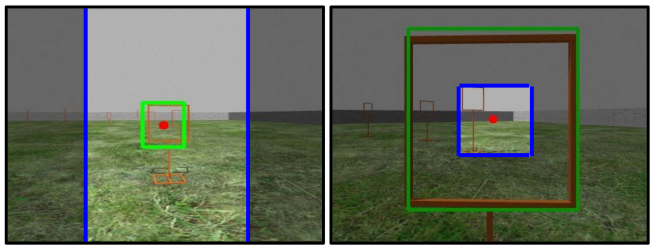


Figure 3: Example of cases to obtain the value of k_θ. (a) If the detection is within the boundary indicated by blue lines, the value of k_θ is obtained using Equation 2. (b) if the centroid of the detection is within the blue box, the gains are equal to $k_\theta = 2$, $k_{\phi=0}$ and $k_{\psi=0}$, which seeks to accelerate the forward motion twice the value estimated by DeepPilot for S_θ

.

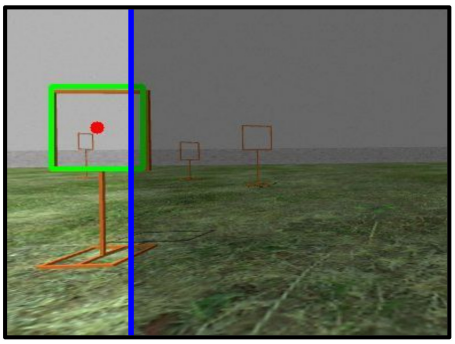


Figure 4: Example of when the gate is near the edge of the image, DeepPilot assigned $S_\phi = 0$ even though the gate is inside the viewing area. For this reason, we replaced the flight signal S_ϕ by 1 or -1, depending on what edge the gate is located, to force the drone to respond in the direction of detection.

time moving from one side to the other, ultimately colliding with the gate edge, or flying outside the gate. We mitigate this by defining k_ϕ with Equation 3, where d is the distance in pixels between the centroid of the detection and the centre of the image, as we showed in Figure 4. Note that d is normalised to half the image's width. Thus the value of k_ϕ is proportional to d, which helps reduce the roll signal provided by DeepPilot.

$$k_\phi = \frac{d}{320} \tag{3}$$

The last case is the slow rotation when the gates are skewed, as shown in Figure 6. We obtain the value of k_ψ using Equation 4 to adjust the rotation signal depending on the width of the bounding box detecting the gate. Here, w is the width of gate detection, which is subtracted from 1 and normalised by the height to obtain an inverse ratio to w.

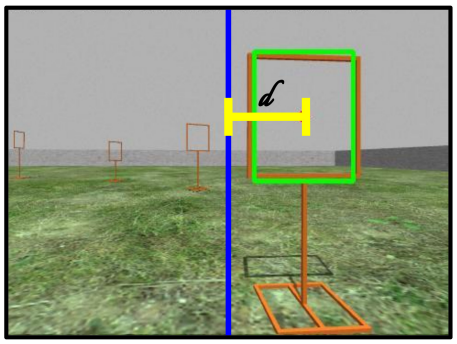


Figure 5: Case 3 used to automatically tune the gain for the roll signal. To this purpose, the distance in pixels from the centroid of the detection and the centre of the image is calculated. Half the image width normalises this distance, giving a ratio that reduces the value of the roll signal provided by DeepPilot. This variable gain helps to reduce oscillations when the drone attempts to centre w.r.t. to the gate.

$$k_\psi = \frac{1.0 - w}{640} \tag{4}$$

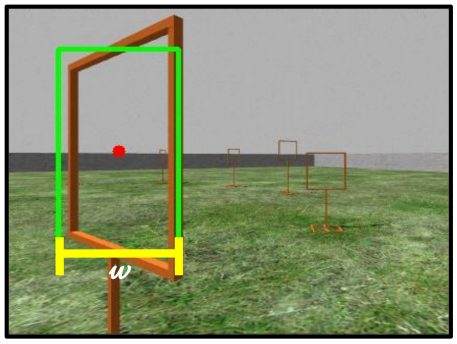


Figure 6: Automatic tuning for the yaw signal under skew views of the gate. This case occurs in turns of the track, mainly due to the proximity between one gate and another. Due to the drone's inertia, the camera has an inclined view of the gate, thus producing a skew view of the gate. Even when DeepPilot estimates a yaw signal, its value may not be enough to align the drone on time. This is overcome with the automatic gain tuning for this case when setting the gain as an inverse function of the width of the bounding box of the detected gate, as shown in Equation 4.

4 EXPERIMENTS

4.1 System Overview

Our proposal was implemented on the Alienware R5 laptop, which has a corei7 processor, 32GB of RAM and an NVIDIA GTX 1070 graphics card. It runs the Ubuntu 18.04 LTs operating system and ROS Melodic Morenia. We used

http://www.imavs.org/

the Gazebo 9 simulator and the rotors_simulator package to emulate the Parrot Bebop 2 vehicle for the experimental framework; we should highlight that in RotorS[7], the Bebop 2 model exhibits a more agile flight (closer to reality) than that used in the TUM simulator [23]. For the execution of the SSD7 and DeepPilot networks, we used the TensorFlow 1.15 and Keras 2.3.1 frameworks.

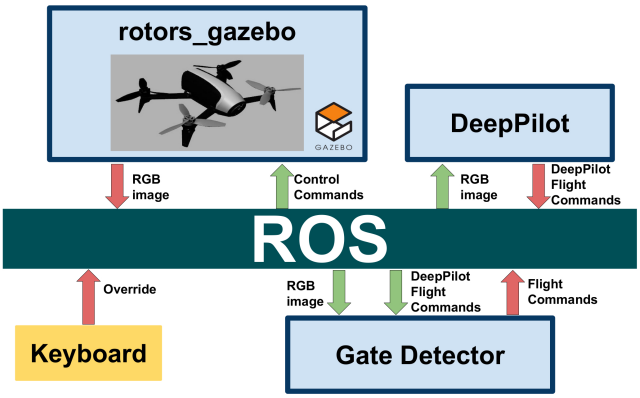


Figure 7: Communication system based on the Robotic Operating System. Green arrows indicate data published by the nodes. Red arrows indicate data consumed by other nodes.

Figure 7 shows the communication architecture used in this work. It shows the interaction between the ROS nodes. Where the principal node is rotors_simulators, which provides information from the camera onboard the Bebop2 at 80 fps, the odometry and simultaneously receives information to control the Bebop2. The DeepPilot node gets the image from Bebop2, generates an image mosaic updated every five frames and provides four flight signals $(S_\phi, S_\theta, S_\psi, S_h)$ at 25 fps. The Gate Detector node receives the Bebop2 image, and we applied the trained model for gate detection. It also gets the flight signals provided by DeepPilot and publishes the flight signals with the automatic gain tuning at 80 fps. Finally, the node Keyboard is used only to initiate or cancel the autonomous flight.

4.2 DeepPilot vs DeepPilot with fixed gain

We use the evaluation racetrack in [6], which is composed of 18 gates. The racetrack extends over $62m \times 44m$, and the spacing between gates is 7m to 9m. Therefore, gates 1, 2, 3, 5, 9, 12 and 14 are 2m high, gates 4, 7, 8, 11, 13, 15, 16 and 18 are 2.5m high, and gates 6, 10 and 17 are 3m high, all of them are placed in an ellipse shape to vary the orientations.

We performed experiments on the Bebop2 platform of the rotors_gazebo simulator using the model trained on the same track using the Ar drone platform of the tum_simulator simulator[1], see Figure 8, where DeepPilot did not perform well as it failed to cross the 18 gates of the race track. DeepPilot

[1]https://github.com/QuetzalCpp/DeepPilot/tree/Noetic

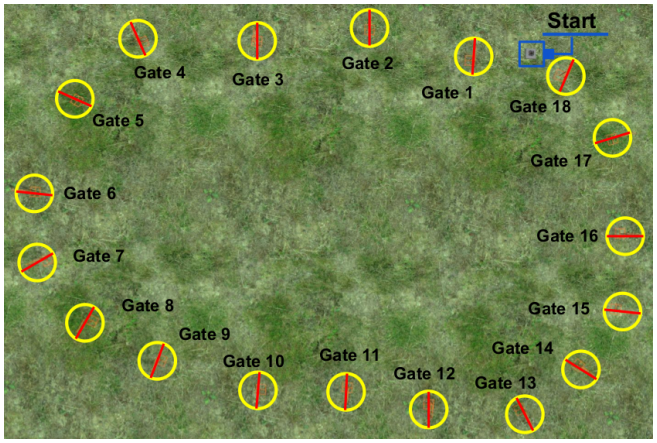


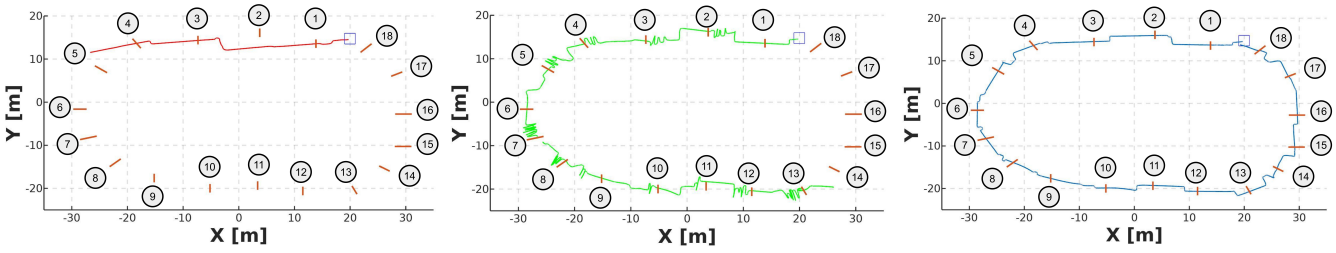
Figure 8: The track consists of 18 gates set at different heights and orientations. The racetrack extends over 62m and has gate spacing of 7m to 9m. Image is taken from [6].

only managed to cross 3 out of 18 gates in ten runs. In this case, the drone always evaded gate number two, as shown in Figure 9(a). Therefore, we analysed the flight signal values obtained during the ten trajectories, where the average is 0.023 for roll, 0.306 for pitch, 0.008 for yaw and 0.006 for altitude. These values indicate that the pitch is too high compared to the rest of the flight signals, which does not allow the drone to cross all the gates in the track.

Therefore, via several flight experiments, we found a set of fixed gains for which, after multiplying by the flight signals estimated by DeepPilot, produced a reasonable performance. This is, the drone managed to cross almost all the gates, although it always hit or flew outside some gates. Thus, with this empirical analysis, we set the gains to be $k_\theta = 0.5$, $k_\phi = 1$, $k_\psi = 5$ and $k_h = 1.2$.

The fixed gains enabled the drone to cross 12 of 18 gates, with gate number 11 being evaded. This can be appreciated in Figure 9(b), where we can see that the drone positioned itself in front of the gate throughout the trajectory. However, the variation of values generated by the roll signal causes an oscillation that stops the drone from crossing the gate. This can be seen in 2 cases, the first at gate 11, where it evades to the left and does not cross the gate, and the second at gate 14, since due to the oscillations the drone cannot correctly generate a signal to yaw. We performed 10 runs using these fixed gains to obtain the following average flight signal values: 0.020 in roll, 0.095 in pitch, 0.012 in yaw and 0.004 in altitude. These values indicate that small values of the pitch signal improves DeepPilot's performance in guiding the drone on the racetrack. However, the oscillations of the roll signal do not allow the drone to complete the track.

http://www.imavs.org/



(a) (b) (c)

Figure 9: Top view of 1 out of 10 runs in our experiments: (a) corresponds to the drone's trajectory using directly the flight signals provided by DeepPilot; (b) corresponds to the drone's trajectory using the flight signal provided by DeepPilot multiplied by fixed gains set up manually; (c) corresponds to the drone's trajectory using the flight signals provided by DeepPilot multiplied with gains tuned automatically by using gate detection, note that in this case the drone is able to traverse all the gates and finish the lap. A video illustrating these experiments can be seen at `https://youtu.be/49OzJCHTJu4`



(a) (b) (c)

Figure 10: Examples of DeepPilot failing when the gate is at the edge of the image. The top images are snapshots of the 6-image mosaic that DeepPilot uses to estimate flight signals. The inferior pictures are snapshots of the drone and the gate in the gazebo world.

4.3 *DeepPilot with automatic gain tuning using gate detection*

Due to the instability of the drone to complete the race-track, we employed the gate detection for automatic gain tuning under the rules set out in Section 3.3. DeepPilot operates using the $k_\theta = 0.5$, $k_\phi = 1$, $k_\psi = 5$ and $k_h = 1.2$ as default values but these are modified when the SSD7 network detects the nearest gate. Only then do the gains change their value to modify the drone's behaviour guided by DeepPilot.

In Figure 9(c), we note that case 1 is activated on gates 3-4, 5-6, 8-9, 9-10 and 16-17; for instance, the value of k_θ increases, as the gates are far away from each other. Case 2 is activated at the exit of gates 4, 13, 14 and 18, as the neighbouring gates are on a curve. By changing the roll value, the drone is forced to continue on its way. This failure by DeepPilot is attributed to the lack of examples for this type of view. The effectiveness of case 3 is noticeable along the track since the trajectory does not show large oscillations compared to the performance of DeepPilot with fixed gains, see Figure 9(b). Finally, case 4 is activated at gates 8, 14 and 18, as the previous gates are close, exhibiting skew views. In this case, the rule using the the gate detection helps to increase the yaw signal in order to align the drone parallel to the gate.

In Table 1, we compare the performance of DeepPilot against DeepPilot with fixed gains and DeepPilot with automatic gain tuning using gate detection. Note that DeepPilot, with no gains, failed to complete the track, and this is because, on average, the pitch value is higher than the other

http://www.imavs.org/

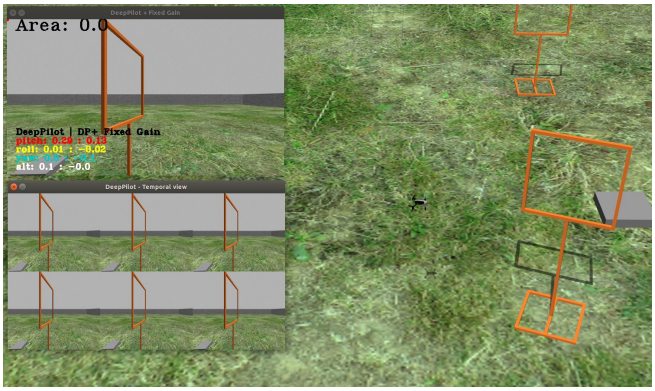


Figure 11: Examples of DeepPilot failing when the gate exhibits a skew view.

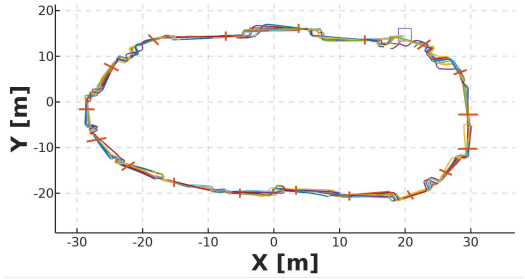
flight signals, making it impossible for the drone slow down and allow the necessary time for the other signals to get estimated properly. When using fixed gains, DeepPilot can cross up to 12 gates.

Nevertheless, the roll oscillations make it fail to cross all the gates. In addition, DeepPilot fails in cases where the gate is at the edge of the image, see Figure 10 or with skew views, as shows Figure 11. Thus, these experiments show the benefits of the automatic gain tuning, which enable the drone to finish the lap. Note that the average pitch is high, meaning that the automatic tuning is effective to assign a high gain value in those cases when the gate is far away or close and centred on the image.

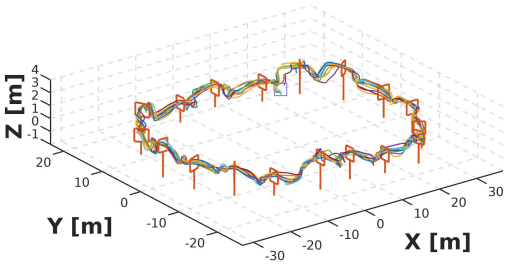
For a time comparison and given that DeepPilot with fixed gains failed after crossing gate 13, the last column in Table 1 shows the average flight time up to gate 13 for DeepPilot with fixed gains and DeepPilot with automatic tuning. Note that the former took a longer time to reach gate 13. This indicates that the automatic gain tuning also helped to reduced the flight time.

A video illustrating these experiments can be seen at `https://youtu.be/49OzJCHTJu4`

In Figure 12(a), we show the performance of DeepPilot with automatic gain tuning using gate detection in 10 runs. We can see that the drone behaves in a stable and repetitive manner. Minimal oscillations in altitude are observed, see Figure 12(b), where the average of the flight signals values is: 0.007 in roll, 0.211 in pitch, 0.038 in yaw and 0.007 in altitude. Finally, Figure 13 shows that our proposal generates a stable and consistent flight that kept the drone flying during 3 hours, performing 18 alps without colliding with the gates, thus obtaining an average time of 9.81 minutes per lap. In these consecutive flights, the average value of the flight signal is 0.004 in roll, 0.217 in pitch, 0.041 in yaw and 0.001 in altitude. This indicates that DeepPilot + GateDet got its flight signals adjusted, avoiding abrupt changes during each lap. Note that this behaviour resembles that of a human pi-



(a)



(b)

Figure 12: Top and side view of the trajectories for 10 runs using the flight signals provided by DeepPilot with automatic gain tuning using gate detection. Note that the drone behaves stably and repetitively. Minimal oscillations in altitude are observed.

lot who improves his performance after each lap. This is an interesting phenomenon that we will analyse in further study.

Finally, Figure 14 shows two runs using DeepPilot automatic gain tuning in two scenarios. In the first one, shown in light blue, it performs the flight with fixed position of the gates in the race track. For the second scenario, indicated in orange, gates 1, 2, 3 change position three times, 5, 8, 10 twice, 11, 12, 16 and 17 change only once. These variations are indicated with dark blue lines. This experiments demonstrates that our proposal works effectively under dynamic changes in the race track.

5 Conclusion

We have presented an automatic gain tuning approach for a neural pilot called DeepPilot developed for Autonomous Drone Racing. DeepPilot estimates flight signals to enable the drone to cross the gates in the race track. However, the estimated flight signals should be noise filters in output values to smooth out peaks in the flight signals, which produce oscillatory behaviour and jolts. Therefore, in this work, we proposed using a gate detector to tune such gains automatically. Usually, a gate detector in ADR has been used to implement a flight controller. However, the detector does not provide 3D information about the gate; thus, such a controller could not be used to control the yaw signal when a gate is observed in a skew view. In contrast, DeepPilot has learned to estimate a

http://www.imavs.org/

Table 1: Comparison of the performance of DeepPilot against DeepPilot with fixed gains and DeepPilot with adjustable gains using gate detection in a race track of 18 gates. The values $(S_\phi, S_\theta, S_\psi, S_h)$ correspond to the average of flight signals sent to the drone over ten trajectories. Also, we report the number of gates crossed in ten runs and the average flight time up to gate 13 for comparison. Note that the automatic gain tuning helps to reduce the flight time in a half.

Method	Gains	Avg. Flight Signal				Avg. Crossed Gates	Avg. Time up to Gate 13 [min.]
		S_ϕ	S_θ	S_ψ	S_h		
DeepPilot	-	0.023	0.306	0.008	0.006	3/18	-
DeepPilot	Fixed	0.020	0.095	0.012	0.004	12/18	15.20
DeepPilot + Gate Det	Automatic Tuning	0.007	0.211	0.038	0.007	**18**/18	**7.15**

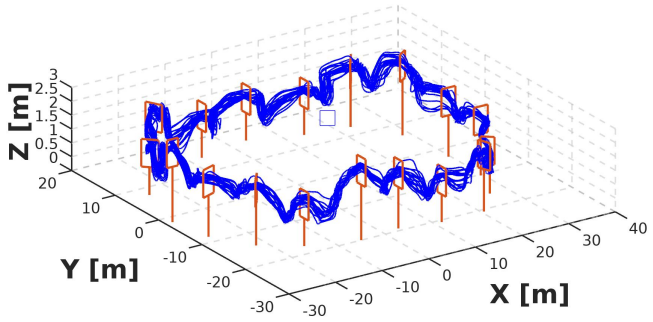


Figure 13: Trajectories of 18 consecutive flights without landing between laps. The drone uses the flight signals provided by DeepPilot multiplied with gains tuned automatically by using gate detection.

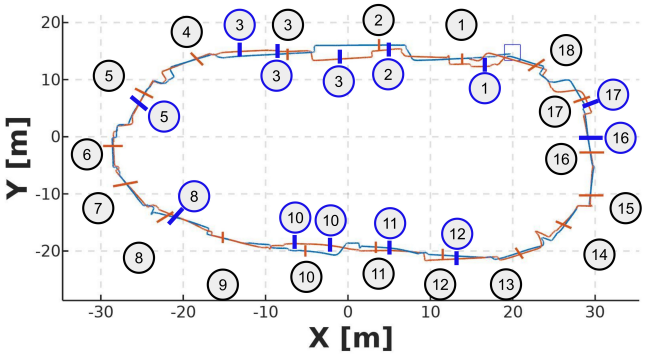


Figure 14: Example of two runs using DeepPilot with automatic gain tuning using gate detection. No changes are made to the gate positions in the light blue trajectory. While in the orange trajectory, the gates are moved along the path. The dark blue circles indicate the number of gates that were moved, and dark blue lines indicate the position to which each of the gates moved. A video illustrating this experiments can be seen at `https://youtu.be/49OzJCHTJu4`

corresponding yaw signal in such a scenario. Therefore, we have proposed to use the gate detector to leverage the performance of DeepPilot by automatically tuning its gains depending upon the gate position on the image. Our experiments in simulation have shown that our proposed approach achieves better results than the original DeepPilot in a large race track with 18 gates randomly placed with variations in orientation and height. For our future work, we will carry out experiments in real scenarios.

References

[1] 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.

[2] Philipp Foehn, Dario Brescianini, Elia Kaufmann, Titus Cieslewski, Mathias Gehrig, Manasi Muglikar, and Davide Scaramuzza. Alphapilot: Autonomous drone racing. *arXiv preprint arXiv:2005.12813*, 2020.

[3] Christophe De Wagter, Federico Paredes-Vallés, Nilay Sheth, and Guido de Croon. The artificial intelligence behind the winning entry to the 2019 ai robotic racing competition. *arXiv preprint arXiv:2109.14985*, 2021.

[4] Philipp Foehn, Angel Romero, and Davide Scaramuzza. Time-optimal planning for quadrotor waypoint flight. *Science Robotics*, 6(56):eabh1221, 2021.

[5] Rogerio Bonatti, Ratnesh Madaan, Vibhav Vineet, Sebastian Scherer, and Ashish Kapoor. Learning visuomotor policies for aerial navigation using cross-modal representations. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1637–1644. IEEE, 2020.

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

[7] Fadri Furrer, Michael Burri, Markus Achtelik, and Roland Siegwart. *Robot Operating System (ROS): The Complete Reference (Volume 1)*, chapter RotorS—A

http://www.imavs.org/

Modular Gazebo MAV Simulator Framework, pages 595–625. Springer International Publishing, Cham, 2016.

[8] 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.

[9] 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.

[10] Aldrich A Cabrera-Ponce, Leticia 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.

[11] S Li, C De Wagter, CC de Visser, QP Chu, GCHE de Croon, et al. In-flight model parameter and state estimation using gradient descent for high-speed flight. *International Journal of Micro Air Vehicles*, 11:1756829319833685, 2019.

[12] 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.

[13] Shuo Li, Erik van der Horst, Philipp Duernay, Christophe De Wagter, Guido CHE de Croon, et al. Visual model-predictive localization for computationally efficient autonomous racing of a 72-g drone. *Journal of Field Robotics*, 2020.

[14] Shuo Li, Michaël 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*, 133:103621, 2020.

[15] 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.

[16] Elia Kaufmann, Mathias Gehrig, Philipp Foehn, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Beauty and the beast: Optimal methods meet learning for drone racing. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 690–696. IEEE, 2019.

[17] 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.

[18] Elia Kaufmann, Antonio Loquercio, Rene Ranftl, Alexey Dosovitskiy, Vladlen Koltun, Davide Scaramuzza, et al. Deep drone racing: Learning agile flight in dynamic environments. In *Conference on Robot Learning*, pages 133–145. PMLR, 2018.

[19] Jos Arturo Cocoma-Ortega and J Martinez-Carranza. A cnn based drone localisation approach for autonomous drone racing. In *11th International Micro Air Vehicle Competition and Conference*, 2019.

[20] José Arturo Cocoma-Ortega and José Martínez-Carranza. Towards high-speed localisation for autonomous drone racing. In *Mexican International Conference on Artificial Intelligence*, pages 740–751. Springer, 2019.

[21] Christian Pfeiffer and Davide Scaramuzza. Expertise affects drone racing performance. *arXiv preprint arXiv:2109.07307*, 2021.

[22] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[23] H Huang and J Sturm. Tum simulator, 2014.

http://www.imavs.org/

INDI Control for the Oblique Wing-Quad Plane Drone

D.C. van Wijngaarden*, and B.D.W. Remes
Delft University of Technology, Kluyverweg 1, Delft

ABSTRACT

A type of UAV sharing the advantages of rotorcraft and fixed wing vehicles is the hybrid vehicle. Hybrid UAVs can take-off and land vertically and fly fast and efficient in forward flight due to the presence of a wing generating lift. However, combining the fixed wing and rotorcraft concept ends up in a multi-rotor with large dimensions when landed, and catching a lot of gust when hovering. Therefore, the oblique wing-quad plane drone has been developed which can rotate its quad arm and wing such that the wing can be aligned with the fuselage in hover. The wing can be rotated to fixed wing position during fast forward flight, stowing away one quad-arm. An INDI inner and outer loop controller has been implemented for this platform. Successful flight tests proved the feasibility of this controller and drone concept which is patent pending NL 2031701, Aeronautical Vehicle and Method of Transitioning between Flight Modes for an Aeronautical Vehicle, April 26th 2022,

http://www.imavs.org/

1 INTRODUCTION

Since 2010 different VTOL-winged drones and control strategies are designed and integrated into different open-source autopilots, resulting in different VTOL- winged drone concepts. Merging multi-rotors and tail-sitters together, resulting in concepts like the Quadshot [1], ATMOS [2] and the NederDrone [3]. Merging a helicopter with a tail-sitter resulted in the DelftAcopter [4] and merging a fixed wing with a multi-rotor give the well know quad-plane concept [5]. Many more concepts have been designed but non of them are optimised for VTOL-winged flight and landing in a confined space like a drone-box.

This paper describes a new drone concept: The Oblique Wing-Quad Plane Drone which has the ability to vertically take-off and land, but the platform can skew its wing and quad arm in order to land in a confined space. During take-off and landing, the wing is directed parallel with the fuselage. While flying at higher airspeed, the wing and quad-arm can be rotated in-flight such that the hover motors will end up being rotated inside the fuselage and the wing being directed with its leading edge in the airflow. This way, a platform can be optimised for forward flight while having the ability to take-off and land in the confined space of a drone-box.

An Incremental Non-linear Dynamic Inversion (INDI) controller is implemented for this drone as inner loop controller [6, 7]. No accurate model of the drone concept is needed for an INDI controller as the control method is sensor based and only actuator effectiveness values and dynamics are necessary to set-up a controller. Furthermore, the performance of disturbance rejection is high due to the direct angular acceleration feedback in the INDI controller. A Weighted Least Square (WLS) control allocation method has been applied to handle actuator saturation [8].

An INDI guidance controller has been derived for the outer loop as well. A similar INDI guidance method has already been proven to work well [9]. A WLS control allocation method was also been applied to the outer loop controller as the drone is over-actuated in pitch [5]. Therefore, a preferred pitch angle can be commanded within the saturation limits of the drone.

The architecture of the oblique wing-quad plane drone is outlined by section 2. The inner and outer loop INDI controllers applied to the drone are described by sections 3 and 4 respectively. A discussion on the results regarding experiments carried out with the drone are discussed in section 5. Finally, this paper end with a conclusion and recommendations for future research as given by section 6.

2 THE OBLIQUE WING-QUAD PLANE DRONE

The Drone-in-a-box concept is being deployed globally, making the drone sector more scalable than ever before. Operating VTOL-winged drones from a box implies limitations. They need a very large box, or they can only operate in low wind / no gust condition, as the wing introduce large disturbances in to the hovering platform due to wind gust.

When flying in fixed wing mode, the VTOL-winged drones actuators optimised for hover flight are often exposed to the free stream air, introducing parasitic drag during forward flight. An ideal VTOL-winged drone-in-a-box has no wing exposed during hover and has maximum multi-rotor control authority, while during cruise flight it has no hover actuators exposed and has a cruise speed optimised wing. Those conflicting demands resulting in the design of the oblique wing-quad plane drone. The drone takes-off, lands and hovers as a tailed quad-copter with an extra pusher propeller as depicted in figure 1. The skew angle Λ of the wing and the quad arm in this state is 0 degrees. When the airspeed increases, one quad arm rotates 90 deg around its midpoint inside the fuselage. While the wing which was aligned with the

*Email address: D.C.vanWijngaarden@tudelft.nl

fuselage during hover, rotates 90 deg, transforming the quad into an efficient fixed wing airplane with a dedicated forward propeller on its tail. The fixed wing state of the drone is given by figure 2. The skew angle Λ is set to 90 degrees in this state.

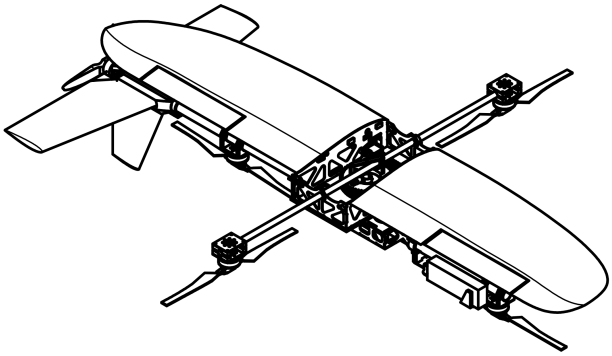


Figure 1: Oblique wing-quad plane with skew angle Λ set to 0 degrees.

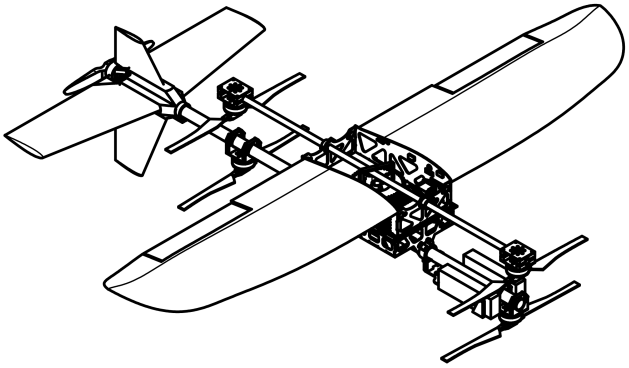


Figure 2: Oblique wing-quad plane with skew angle Λ set to 90 degrees.

The oblique wing-quad plane drone has a total of 10 actuators to control its attitude, linear accelerations and the skew angle of the rotating wing and quad arm. The drone has 5 motors, 4 control surfaces and one servo to rotate one quad arm and the wing. Figure 3 shows a graphical representation of the drone for which the wing is skewed halfway. Each actuator is numbered in the figure and described by table 1 accordingly. The right column of the table shows whether the actuator is rotating with skew angle.

To maximise control authority of the vehicle during hover, fixed wing and transition, a dedicated Incremental Non linear Dynamic Inversion controller (INDI) is designed as described by the next sections in this paper.

3 INDI INNER LOOP CONTROL AND SCHEDULING

The attitude control approach that has been implemented on the rotating wing drone is based on Incremental Nonlinear Dynamic Inversion (INDI) as outlined by [6]. A weighted least square (WLS) control allocation method has been implemented to the INDI attitude controller to prioritise the control of a certain control axis in case of actuator saturation [8]. The WLS method can also prefer the use aerodynamic surfaces over motors to stabilise the drone.

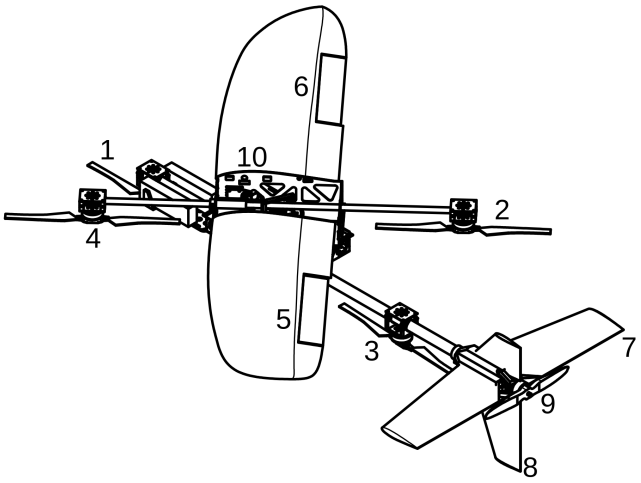


Figure 3: Actuator schematic of the oblique wing-quad plane drone.

Actuator number	Actuator name	Rotating
1	front motor	
2	right motor	✓
3	back motor	
4	left motor	✓
5	left aileron	✓
6	right aileron	✓
7	elevator	
8	rudder	
9	push motor	
10	wing rotation servo	

Table 1: List of actuators for the oblique wing-quad plane drone.

The implemented INDI controller is based on the second law of Newton:

$$\ddot{\boldsymbol{\xi}} = \boldsymbol{g} + m^{-1}\left(\boldsymbol{M}_{NB}\boldsymbol{f} + \boldsymbol{f}_{ext}\right) \tag{1}$$

where $\boldsymbol{\xi}$ denotes the position vector in the North East Down (NED) reference frame and $\boldsymbol{g}$ the gravitational acceleration in the same reference frame. The value of m is the mass of the drone. The force vector $\boldsymbol{f}$ is the input force in the body reference frame due to changes in thrust level and attitude. $\boldsymbol{M}_{NB}$ is the rotation matrix from the body frame to the NED reference frame. The matrix is obtained from the at-

http://www.imavs.org/

titude quaternion $\boldsymbol{q}$. Finally, vector $\boldsymbol{f}_{ext}$ contains the external unmodeled forces acting on the system.

The equation of rotational dynamics of the system is given by:

$$\dot{\boldsymbol{\Omega}} = \boldsymbol{J}^{-1}(\boldsymbol{m} + \boldsymbol{m}_{\text{ext}} - \boldsymbol{\Omega} \times \boldsymbol{J}\boldsymbol{\Omega}), \tag{2}$$

where $\boldsymbol{\Omega}$ is the attitude vector containing the roll, pitch and yaw angle respectively. The vector $\boldsymbol{J}$ is the vehicle's mass moment of inertia in the body reference frame. The vectors $\boldsymbol{m}$ and $\boldsymbol{m}_{ext}$ represent the moments generated by control inputs and unmodeled external disturbances respectively.

It can be approximated that the change in angular acceleration of the system is induced due to a change in input vector $\boldsymbol{u}$ by neglecting cross terms in the inertia matrix $\boldsymbol{J}$:

$$\dot{\boldsymbol{\Omega}} - \dot{\boldsymbol{\Omega}}_0 = \boldsymbol{G}_1(\boldsymbol{u} - \boldsymbol{u}_0) + \boldsymbol{G}_2(\dot{\boldsymbol{u}} - \dot{\boldsymbol{u}}_0), \tag{3}$$

where $\boldsymbol{G}_1$ is the control effectiveness matrix and $\boldsymbol{G}_2$ is the control effectiveness with respect to the derivative of the control input vector $\boldsymbol{u}$.

The first 8 actuators given in table 1 can manipulate the angular accelerations as given by $\boldsymbol{\Omega}$. Therefore, the INDI inner loop controls those 8 actuators and therefore the control input vector $\boldsymbol{u}$ consist of 8 entries. As a result, the effectiveness matrices $\boldsymbol{G_1}$ and $\boldsymbol{G_2}$ consists of 8 columns and 3 rows. The effectiveness matrix $\boldsymbol{G}_1$ is build up as follows:

$$\boldsymbol{G}_1 = \begin{bmatrix} \boldsymbol{G_{1_1}} & \boldsymbol{G_{1_2}} & \boldsymbol{G_{1_3}} & \boldsymbol{G_{1_4}} & \boldsymbol{G_{1_5}} & \boldsymbol{G_{1_6}} & \boldsymbol{G_{1_7}} & \boldsymbol{G_{1_8}} \end{bmatrix} \tag{4}$$

In the first place, control effectiveness values need to be scheduled with the drone's skew angle Λ as some actuators are moving with changing skew angle as shown by table 1. This scheduling is applied to the side-motors and ailerons. A 3D CAD model shows the roll and pitch moments of inertia only marginally change by changing the skew angle. Therefore, the pitch and roll control effectiveness values of the side motors and ailerons can be scheduled with trigonometric functions of the skew angle Λ.

Secondly, the control effectiveness of aerodynamic surfaces are scheduled with dynamic pressure. Those effectiveness values are therefore dependent on the airspeed squared. Airspeed is measured by a pitot tube directed in the body x axis.

The first column of the effectiveness matrix $\boldsymbol{G}_1$ corresponds to the front motor which can control angular pitch and yaw accelerations as shown by the following:

$$\boldsymbol{G}_{1_1} = \begin{bmatrix} 0 & g_{1_{21}} & g_{1_{31}} \end{bmatrix}^T \tag{5}$$

The second column of the control effectiveness matrix corresponds to the right motor for which roll and pitch effectiveness is scheduled by the cosine and sine of the skew angle Λ respectively. The roll effectiveness was determined for a skew angle Λ of 0 degrees, quad-mode, and subsequently scheduled by scaling it with the cosine of the skew angle. On the contrary, the pitch effectiveness has been determined for the fixed wing state with skew angle Λ set to 90 degrees and thereafter scheduled by scaling this value with the sine of the skew angle as shown below:

$$\boldsymbol{G}_{1_2} = \begin{bmatrix} g_{1_{12}}^{\Lambda=0}\mathrm{c}\Lambda & g_{1_{22}}^{\Lambda=\pi/2}\mathrm{s}\Lambda & g_{1_{32}} \end{bmatrix}^T \tag{6}$$

The third column of the effectiveness matrix corresponds to the back motor which is equivalent to the front motor except the sign of the pitch effectiveness value:

$$\boldsymbol{G}_{1_3} = \begin{bmatrix} 0 & g_{1_{23}} & g_{1_{33}} \end{bmatrix}^T \tag{7}$$

The control effectiveness scheduling of the left motor is equivalent to the right motor except the signs of the roll and pitch effectiveness values air opposite. The fourth column of the control effectiveness matrix corresponding to this motor is given below:

$$\boldsymbol{G}_{1_4} = \begin{bmatrix} g_{1_{14}}^{\Lambda=0}\mathrm{c}\Lambda & g_{1_{24}}^{\Lambda=\pi/2}\mathrm{s}\Lambda & g_{1_{34}} \end{bmatrix}^T \tag{8}$$

The roll and pitch effectiveness of the left and right ailerons can be scheduled by sine and cosine functions based on the skew angle Λ as those control surfaces are being displaced by changing the skew angle. In the current research stage, the control effectiveness values of the ailerons in pitch $g_{1_{25}}$ and $g_{1_{26}}$ have not yet been defined and set to 0 such that the ailerons will only handle roll control. The roll effectiveness values $g_{1_{15}}^{\Lambda=\pi/2}$ and $g_{1_{16}}^{\Lambda=\pi/2}$ have been defined in fixed wing state and scheduled by airspeed squared. The effectiveness columns in effectiveness matrix $\boldsymbol{G}_1$ of the left and right ailerons are defined by the following two equations respectively:

$$\boldsymbol{G}_{1_5} = \begin{bmatrix} g_{1_{15}}^{\Lambda=\pi/2}\left(V^2\right)\mathrm{s}\Lambda & g_{1_{25}}^{\Lambda=0}\left(V^2\right)\mathrm{c}\Lambda & 0 \end{bmatrix}^T \tag{9}$$

$$\boldsymbol{G}_{1_6} = \begin{bmatrix} g_{1_{16}}^{\Lambda=\pi/2}\left(V^2\right)\mathrm{s}\Lambda & g_{1_{26}}^{\Lambda=0}\left(V^2\right)\mathrm{c}\Lambda & 0 \end{bmatrix}^T \tag{10}$$

The elevator effectiveness corresponds to the the seventh column and is only effective in pitch. As this actuator is a control surface, it is aerodynamically being scheduled by airspeed squared as shown below:

$$\boldsymbol{G}_{1_7} = \begin{bmatrix} 0 & g_{1_{27}}\left(V^2\right) & 0 \end{bmatrix}^T \tag{11}$$

Finally, the last column of the effectiveness matrix corresponds to the rudder. The surface is only effective in yaw and scheduled with airspeed squared:

$$\boldsymbol{G}_{1_8} = \begin{bmatrix} 0 & 0 & g_{1_{38}}\left(V^2\right) \end{bmatrix}^T \tag{12}$$

http://www.imavs.org/

For the $\boldsymbol{G}_2$ effectiveness matrix it is assumed that only the first four actuators (hover motors) generate a moment in yaw with respect to the derivative of the control input vector $\boldsymbol{u}$.

For INDI control, equation 3 should be discretized and subsequently inverted in order to compute the control input vector $\boldsymbol{u}$ at every time step of the control loop. The values of the angular accelerations $\dot{\boldsymbol{\Omega}}_0$ can be obtained by differentiation of the onboard gyroscope reading. Due to the presence of high frequency vibrations, a low-pass filter is applied to the angular accelerations denoted by subscript f. The control input vector $\boldsymbol{u}_0$ is filtered by the same low-pass filter in order to synchronise the input with the filtered angular acceleration signal. The value of $\dot{\boldsymbol{u}}$ can approximated using the unit delay operator L. A prediction of the angular accelerations $\dot{\boldsymbol{\Omega}}$ can be made based on a new control input command $\boldsymbol{u}_c$:

$$\dot{\boldsymbol{\Omega}} - \dot{\boldsymbol{\Omega}}_f = (\boldsymbol{G}_1 + \boldsymbol{G}_2)(\boldsymbol{u}_c - \boldsymbol{u}_f) + \boldsymbol{G}_2 L(\boldsymbol{u}_c - \boldsymbol{u}_f). \quad (13)$$

In order to compute the control input $\boldsymbol{u}_c$, the equation can be inverted by a pseudo inverse. However, the pseudo inverse of the effectiveness matrices is non singular due to the over-actuated characteristic of the oblique wing-quad plane drone. For that reason, a sophisticated Weighted Least Square (WLS) control allocation method is applied to the inner loop [8]. Inverting equation 13 yields:

$$\boldsymbol{u}_c = \boldsymbol{u}_f + (\boldsymbol{G}_1 + \boldsymbol{G}_2)^{+}(\boldsymbol{\nu} - \dot{\boldsymbol{\Omega}}_f + \boldsymbol{G}_2 z^{-1}(\boldsymbol{u}_c - \boldsymbol{u}_f)) \quad (14)$$

Where the $(.)^{+}$ denotes the inverse which is being handled by the applied WLS method. The vector $\boldsymbol{\nu}$ reference angular accelerations that are simply be calculated using a PD controller. Due to direct feedback of the angular accelerations by differentiating the gyroscope, disturbances can be counteracted fast [6, 7].

4 INDI OUTER LOOP CONTROL

The INDI outer loop controller of the oblique wing-quad plane drone provides the method for guidance for the platform. Derivations of an INDI outer loop controller for multiple types of drones is described by [9, 10]. For over-actuated systems such as the oblique wing-quad plane, a control allocation method can be applied on the outer loop as applied in [5].

The outer loop INDI controller is based on the second law of Newton as shown below:

$$\ddot{\boldsymbol{\xi}} = \boldsymbol{g} + \frac{1}{m}\boldsymbol{L}_N(\boldsymbol{\eta}, V, \Lambda) + \frac{1}{m}\boldsymbol{D}_N(\boldsymbol{\eta}, V, \Lambda) + \frac{1}{m}\boldsymbol{T}_N(\boldsymbol{\eta}, T) \quad (15)$$

The vector $\ddot{\boldsymbol{\xi}}$ consists of the vehicle's accelerations in the NED reference system. The vector $\boldsymbol{g}$ is the gravitational acceleration which is directed downwards. The vectors $\boldsymbol{L}_N$, $\boldsymbol{D}_N$ and $\boldsymbol{T}_N$ represent the lift, drag and thrust forces in the NED reference frame respectively. Vector $\boldsymbol{\eta}$ represents the attitude in ZXY rotation order $\left(\boldsymbol{\eta} = \begin{bmatrix} \psi & \phi & \theta \end{bmatrix}\right)$. The lift and drag in the NED frame are dependent on attitude, velocity and skew angle. The thrust force in this frame of reference is dependent on attitude and thrust input to the hover motors and pusher motor.

In order to transform forces generated in the body reference frame to the NED reference frame, the following rotation matrix in ZXY rotation order has been derived:

$$\boldsymbol{M}_{NB} = \begin{bmatrix} \mathrm{c}\theta\mathrm{c}\psi - \mathrm{s}\phi\mathrm{s}\theta\mathrm{s}\psi & -\mathrm{c}\phi\mathrm{s}\psi & \mathrm{s}\theta\mathrm{c}\psi + \mathrm{s}\phi\mathrm{c}\theta\mathrm{s}\psi \\ \mathrm{c}\theta\mathrm{s}\psi + \mathrm{s}\phi\mathrm{s}\theta\mathrm{c}\psi & \mathrm{c}\phi\mathrm{c}\psi & \mathrm{s}\theta\mathrm{s}\psi - \mathrm{s}\phi\mathrm{c}\theta\mathrm{c}\psi \\ -\mathrm{c}\phi\mathrm{s}\theta & \mathrm{s}\phi & \mathrm{c}\phi\mathrm{c}\theta \end{bmatrix} \quad (16)$$

Thrust of the oblique wing-quad plane drone can be generated in two directions in the body frame. The hover props can generate thrust in the negative Z direction (T_{B_z}) using the hover motors and positive thrust in the X direction using the pusher motor (T_{B_x}) as shown below:

$$\boldsymbol{T}_B = \begin{bmatrix} T_{B_x} & 0 & T_{B_z} \end{bmatrix}^T \quad (17)$$

Rotating the body thrust vector $\boldsymbol{T}_B$ to the NED reference system in ZXY order yields the vector $\boldsymbol{T}_N$:

$$\boldsymbol{T}_N = \begin{bmatrix} (\mathrm{c}\theta\mathrm{c}\psi - \mathrm{s}\phi\mathrm{s}\theta\mathrm{s}\psi)T_{Bx} + (\mathrm{s}\theta\mathrm{c}\psi + \mathrm{s}\phi\mathrm{c}\theta\mathrm{s}\psi)T_{Bz} \\ (\mathrm{c}\theta\mathrm{s}\psi + \mathrm{s}\phi\mathrm{s}\theta\mathrm{c}\psi)T_{Bx} + (\mathrm{s}\theta\mathrm{s}\psi - \mathrm{s}\phi\mathrm{c}\theta\mathrm{c}\psi)T_{Bz} \\ -\mathrm{c}\phi\mathrm{s}\theta T_{Bx} + \mathrm{c}\phi\mathrm{c}\theta T_{Bz} \end{bmatrix} \quad (18)$$

The lift force in the body reference frame is directed in the negative Z direction and is dependent on the pitch angle, velocity and skew angle. Assuming small flight path angles, lift can be linearized around a zero pitch angle. Converting the lift to the NED reference system yields $\boldsymbol{L}_N$:

$$\boldsymbol{L}_N = \boldsymbol{M}_{NB}^{\theta=0}\boldsymbol{L}_B(\theta, V, \Lambda) = \begin{bmatrix} \mathrm{s}\phi\mathrm{s}\psi L(\theta, V, \Lambda) \\ -\mathrm{s}\phi\mathrm{c}\psi L(\theta, V, \Lambda) \\ \mathrm{c}\phi L(\theta, V, \Lambda) \end{bmatrix} \quad (19)$$

It is assumed that the drag forces changes slowly with respect to thrust and lift such that the influence on the change in acceleration can be neglected. This means that only thrust and lift forces need to be evaluated for the construction of the outer loop effectiveness matrix. This also means that for straight and level flight at constant speed, the value of $T_{b_x} = 0$ as T_{b_x} is the only modelled force in the body X axis.

An incremental model can be derived using a first order Taylor expansion. This can be done by taking partial derivatives with respect to the controlled input variables $\left(\boldsymbol{v} = \begin{bmatrix}\phi & \theta & a_{B_z} & a_{B_x}\end{bmatrix}^T\right)$ as shown in the below equation:

$$\ddot{\boldsymbol{\xi}} = \ddot{\boldsymbol{\xi}}_0 + \frac{1}{m}\left(\boldsymbol{G}_T(\boldsymbol{\eta}, T) + \boldsymbol{G}_L(\boldsymbol{\eta}, V, \Lambda)\right)(\boldsymbol{v} - \boldsymbol{v}_0), \quad (20)$$

http://www.imavs.org/

http://www.imavs.org/

The linear accelerations $\ddot{\boldsymbol{\xi}}_0$ are being measured by the drone's accelerometer. As this sensor also measures high frequency vibrations, this signal should be filtered using a low-pass filter resulting in the filtered acceleration vector $\ddot{\boldsymbol{\xi}}_f$ subscripted with f. Vector $\boldsymbol{v}_0$ is also filtered with the same low-pass filter in order to synchronise the signals of the filtered accelerometer $\ddot{\boldsymbol{\xi}}_f$ with the controlled input variables. After that, the equation can be inverted to obtain:

$$\boldsymbol{v} = \boldsymbol{v}_f + m\left(\boldsymbol{G}_T(\boldsymbol{\eta}, T) + \boldsymbol{G}_L(\boldsymbol{\eta}, V, \Lambda)\right)^{-1}\left(\ddot{\boldsymbol{\xi}}_{\text{ref}} - \ddot{\boldsymbol{\xi}}_f\right) \quad (21)$$

where $\ddot{\boldsymbol{\xi}}_{\text{ref}}$ is the reference acceleration to track which is being generated using a PD loop.

The control effectiveness matrices for thrust and lift are given by:

$$\boldsymbol{G}_T(\boldsymbol{\eta}, T_{Bx}, T_{Bz}) = \begin{bmatrix} \left(\frac{\partial}{\partial\phi}\frac{1}{m}\boldsymbol{T}_N(\phi, \theta_0, \psi_0, T_{Bz0}, T_{Bx0})|_{\phi=\phi_0}\right)^T \\ \left(\frac{\partial}{\partial\theta}\frac{1}{m}\boldsymbol{T}_N(\phi_0, \theta, \psi_0, T_{Bz0}, T_{Bx0})|_{\theta=\theta_0}\right)^T \\ \left(\frac{\partial}{\partial T_{Bz}}\frac{1}{m}\boldsymbol{T}_N(\phi_0, \theta_0, \psi_0, T_{Bz}, T_{Bx0})|_{T_{Bz}=T_{Bz_0}}\right)^T \\ \left(\frac{\partial}{\partial T_{Bx}}\frac{1}{m}\boldsymbol{T}_N(\phi_0, \theta_0, \psi_0, T_{Bz0}, T_{Bx})|_{T_{Bx}=T_{Bx_0}}\right)^T \end{bmatrix}^T \quad (22)$$

and

$$\boldsymbol{G}_L(\boldsymbol{\eta}, V, \Lambda) = \begin{bmatrix} \left(\frac{\partial}{\partial\phi}\frac{1}{m}\boldsymbol{L}_N(\phi, \theta_0, \psi_0, V_0, \Lambda_0)|_{\phi=\phi_0}\right)^T \\ \left(\frac{\partial}{\partial\theta}\frac{1}{m}\boldsymbol{L}_N(\phi_0, \theta, \psi_0, V_0, \Lambda_0)|_{\theta=\theta_0}\right)^T \\ (\boldsymbol{0})^T \\ (\boldsymbol{0})^T \end{bmatrix}^T \quad (23)$$

Further elaborating those effectiveness functions result in:

$$\boldsymbol{G}_T(\boldsymbol{\eta}, T_{B_x}, T_{B_z}) = \frac{T_{b_z}}{m}\begin{bmatrix} \mathrm{c}\phi\,\mathrm{c}\theta\,\mathrm{s}\psi & \mathrm{c}\theta\,\mathrm{c}\psi - \mathrm{s}\psi\,\mathrm{s}\theta\,\mathrm{s}\psi & 0 & 0 \\ -\mathrm{c}\phi\,\mathrm{c}\theta\,\mathrm{c}\psi & \mathrm{c}\theta\,\mathrm{s}\psi + \mathrm{s}\phi\,\mathrm{s}\theta\,\mathrm{c}\psi & 0 & 0 \\ -\mathrm{s}\phi\,\mathrm{c}\theta & -\mathrm{c}\phi\,\mathrm{s}\theta & 0 & 0 \end{bmatrix} + \frac{1}{m}\begin{bmatrix} 0 & 0 & \mathrm{s}\theta\,\mathrm{c}\psi + \mathrm{s}\phi\,\mathrm{c}\theta\,\mathrm{s}\psi & \mathrm{c}\theta\,\mathrm{c}\psi - \mathrm{s}\phi\,\mathrm{s}\theta\,\mathrm{s}\psi \\ 0 & 0 & \mathrm{s}\theta\,\mathrm{s}\psi - \mathrm{s}\phi\,\mathrm{c}\theta\,\mathrm{c}\psi & \mathrm{c}\theta\,\mathrm{s}\psi + \mathrm{s}\phi\,\mathrm{s}\theta\,\mathrm{c}\psi \\ 0 & 0 & \mathrm{c}\phi\,\mathrm{c}\theta & -\mathrm{c}\phi\,\mathrm{s}\theta \end{bmatrix} \quad (24)$$

and

$$\boldsymbol{G}_L(\boldsymbol{\eta}, V, \Lambda) = \frac{1}{m}\begin{bmatrix} \mathrm{c}\phi\,\mathrm{s}\psi\,L\left(\theta, V, \Lambda\right) & \mathrm{s}\phi\,\mathrm{s}\psi\,\frac{\partial}{\partial\theta}L\left(\theta, V, \Lambda\right) & 0 & 0 \\ -\mathrm{c}\phi\,\mathrm{c}\psi\,L\left(\theta, V, \Lambda\right) & -\mathrm{c}\psi\,\mathrm{s}\phi\,\frac{\partial}{\partial\theta}L\left(\theta, V, \Lambda\right) & 0 & 0 \\ -\mathrm{s}\phi\,L\left(\theta, V, \Lambda\right) & \mathrm{c}\phi\,\frac{\partial}{\partial\theta}L\left(\theta, V, \Lambda\right) & 0 & 0 \end{bmatrix} \quad (25)$$

The functions $L(\theta, V)$ and $\frac{\partial}{\partial\theta}L(\theta, V)$ still have to be defined for the vehicles with a wing, for a pure rotorcraft they are zero. Recognizing that the lift and T_{Bz} only need to be known to compute the effectiveness of rolling, we assume level flight and a simple relationship with the pitch angle:

$$L(\theta, V, \Lambda) + T_{Bz}(\theta) \approx L(\theta, \Lambda) + T_{Bz}(\theta) = -9.81m \quad (26)$$

For experiments that have been performed, the reference of the yaw angle ψ was set in direction of a windtunnel duct.

Because the pseudo inverse of the outer loop effectiveness matrix is non-singular, a Weighted Least Square (WLS) control allocation method has been applied to calculate the virtual control vector $\boldsymbol{v}$ [8, 5]. As the UAS is over-actuated in pitch, a preferred pitch angle within the saturation limits of the vehicle can be controlled using this control allocation scheme.

5 TEST RESULTS

A transition flight test was performed with the oblique wing-quad plane drone in order to show the working principles of the INDI inner and outer loops and show the feasibility of the design. The tests have been performed in the Open Jet Facility (OJF) of Delft University of Technology [11]. The OJF is a windtunnel facility with a cross section of about 2 by 2 meters. The facility can generate wind speeds up to 35 m/s. The test section is big enough to fly the oblique wing-quad plane drone within the facility to verify the inner and outer loop controllers. The drone is being tracked by an optitrack system and tethered to the ceiling with a rope for safety. The force is being released from the safety rope during flight in order to mimic real flight characteristics. The position accuracy of the vehicle was appropriate to keep it in the test section of the windtunnel during the experiments.

A hover to forward and back to hover transition flight has been performed in the windtunnel. This is done by starting the drone with 0 skew angle in the windtunnel and let it hover autonomous at a fixed position. Photographs of the of a transition from hover to forward flight are given by figure 4. The top image corresponds to hover flight, the middle image to transition flight and the bottom image to forward flight. An airflow of 17 m/s is being generated when the windtunnel is spinned up. This airflow is measured by the pitot tube of the drone as plotted by figure 5. It can be seen from the plot that the airflow is first being increased and at the end of the experiment being decreased.

in order to have a stable hover flight, the drone starts at a 0 skew angle, but the skew angle is automatically being increased to 90 degrees when higher airspeed is being measured. This is done to transform the drone from a quad into a fixed wing to generate lift with its wings with increasing airspeed. A plot of the skew angle over time is given by figure 6. It can be clearly seen when comparing the plot to the airspeed plot that the skew angle is changing with airspeed.

http://www.imavs.org/

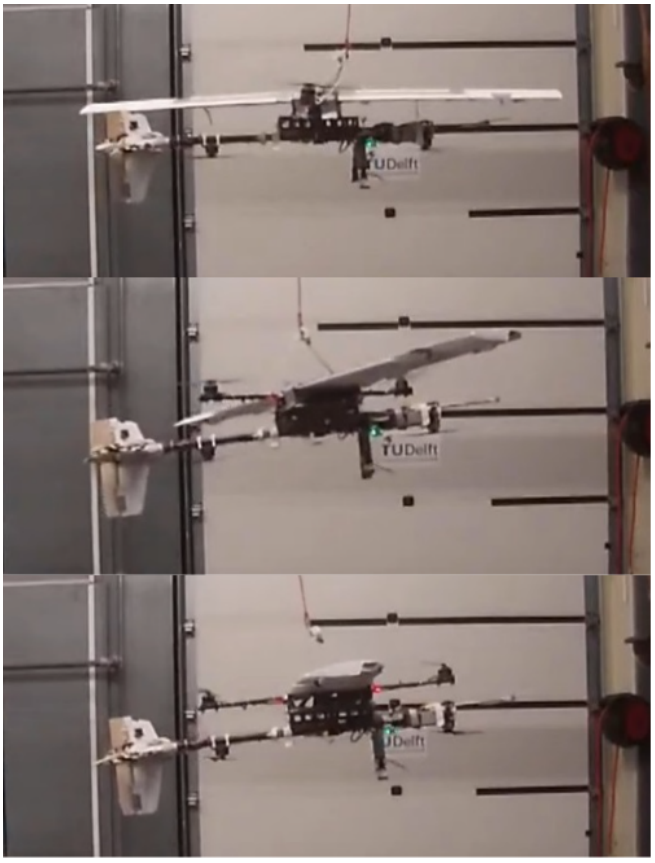


Figure 4: Transition of oblique wing-quad plane drone hover to forward flight.

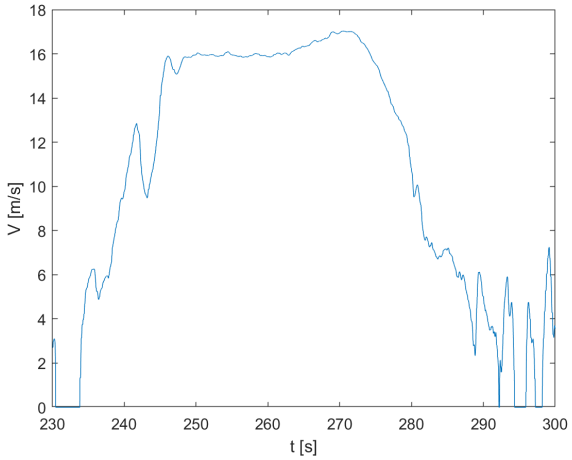


Figure 5: Airspeed versus time during hover-forward-hover transition flight.

Another phenomenon that could be observed from the experiment is the exchange of lift from its hover props (motor 1 to 4) to its wing by looking at the thrust level over time

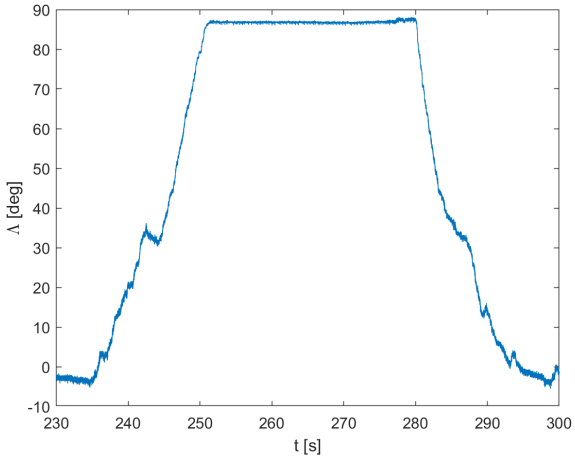


Figure 6: Skew angle versus time during hover-forward-hover transition flight.

plot given by figure 7. The blue line corresponds to the hover motors and it can be clearly seen that thrust is being reduced when airspeed is being increased. On the other hand, the orange line shows that the thrust on the pusher motor is being increased with airspeed as more thrust along the drone's longitudinal axis is required to keep its position in the windtunnel.

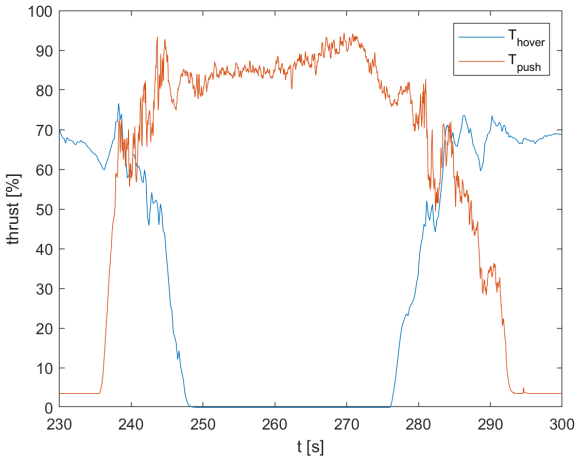


Figure 7: Thrust level versus time during hover-forward-hover transition flight.

The attitude of the drone during the transition test is shown in figure 8. The plot shows the measured and reference angles in blue and orange respectively. From the top and bottom plots, it can be seen that the roll and yaw angles oscillate a little at t=240s and t=280s. This corresponds to a state where the skew angle Λ is set to about 30-45 degrees at an airspeed of about 7-8 m/s. The control authority of the

drone is reduced within this speed regime due to the following phenomena: the back hover motor is blowing air downwards in the vicinity of the elevator. This causes a downward flow of air over the elevator resulting in a negative angle of attack. Therefore, the elevator need to be deflected fully downward up to saturation after which the hover motors saturate in pitch. Due to coupling between the roll, pitch and yaw, this saturation can be seen on other axes as well. This is not a problem for higher airspeed because the thrust generated by the hover motors will gradually go down and the pitch effectiveness of the elevator increases.

Another characteristic of the drone that can be observed from figures 5 to 9 is that the vehicle can fly in fixed wing configuration, even below stall speed, with a skew angle of 90 degrees where upward forces are generated by both the hover motors and wing lift. This can be clearly observed in the time frame running from 275 to 285 seconds. The speed of the airflow is gradually being reduced within this time frame in which the drone starts in fixed wing mode with hover props switched off as can be seen in figure 7. From figure 8 can be observed that first the pitch is being increased after which the hover motors are switched on when the maximum pitch angle has been reached. In this low speed fixed wing state, the hover motors and lift keep the drone up in the air below the stall speed of the fixed wing configuration. It can be observed that from t = 283s the airspeed drops below 8 m/s and only then the wing start to skew back to quad mode as shown in figure 6.

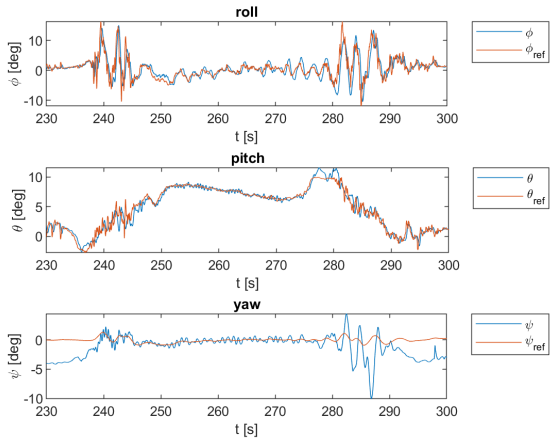


Figure 8: Attitude versus time during hover-forward-hover transition flight.

Finally, a plot of the tracked position and reference position is given by figure 9. It shows the tracked positions with blue lines and reference positions with orange lines. it can be seen that the position error increases for the same time periods as where roll and yaw oscillations occur. The positions errors were small enough to keep the drone during its flight within the test section of the windtunnel without putting any tension on the safety rope.

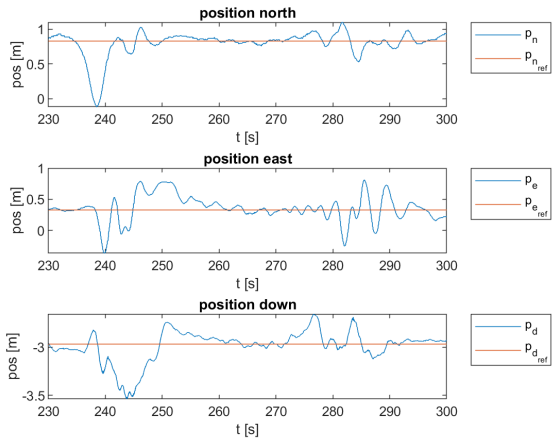


Figure 9: Position versus time during hover-forward-hover transition flight.

6 Conclusions and Recommendations

This paper described the design of the new oblique wing-quad plane drone. It was shown that using the current implemented controller a free flight transition from quad to fixed wing and back could be performed within the 2x2m limits of the test section of the OJF (Open jet windtunnel Facility). We can conclude that the design oblique wing-quad plane is feasible of performing transition in flight within a variety of airspeeds.

The design of the drone can be updated in the future to solve the pitch saturation that occurs around 7-8 m/s airspeed. Also more tests can be performed outdoors and in the windtunnel to be able to model the drone better in order to improve the tuning and effectiveness scheduling of the INDi inner and outer loop control methods. Like the change of the effectiveness of the hover propellers with changing airspeed. Furthermore, an aerodynamic fuselage should be designed to make the drone more efficient in forward flight.

References

[1] P. Sinha, P. Esden-Tempski, C.A. Forrette, J.K. Gibboney, and G.M. Horn. Versatile, modular, extensible vtol aerial platform with autonomous flight mode transitions. pages 1–17, 03 2012.

[2] C. De Wagter, D. Dokter, G.C.H.E. Croon, and B.D.W. Remes. Multi-lifting-device uav autonomous flight at any transition percentage. 01 2013.

[3] C. De Wagter, B.D.W. Remes, R. Ruijsink, F. van Tienen, and E. van der Horst. Design and testing of a vertical take-off and landing uav optimized for carrying

http://www.imavs.org/

a hydrogen fuel cell with a pressure tank. *Unmanned Systems*, 08(04):279–285, 2020.

[4] C. De Wager, B.D.W. Remes, R. Ruijsink, E. Van Der Horst, F. Van Tienen, D.C. Van Wijngaarden, J. Meulenbeld, and K. Van Hecke. DelftaCopter Propulsion Optimization from Hover to Fast Forward Flight using Windtunnel Measurements. *IMAV 2018: International Micro Air Vehicle Conference and Competition 2018*, November:30–38, 2018.

[5] H J Karssies and C. De Wagter. XINCA : Extended Incremental Non-linear Control Allocation on a Quadplane. *IMAV 2021: International Micro Air Vehicle Conference and Competition 2021*, pages 74–84, 2021.

[6] E.J.J. Smeur, Q. Chu, and G.C.H.E. De Croon. Adaptive incremental nonlinear dynamic inversion for attitude control of micro air vehicles. *Journal of Guidance, Control, and Dynamics*, 39(3):450–461, 2016.

[7] Ewoud J. J. Smeur, G.C.H.E. de Croon, and Q. Chu. Cascaded incremental nonlinear dynamic inversion for MAV disturbance rejection. *Control Engineering Practice*, 73:79–90, apr 2018.

[8] E.J.J. Smeur, D.C. Höppener, and C. De Wagter. Prioritized Control Allocation for Quadrotors Subject to Saturation. *IMAV 2017: International Micro Air Vehicle Conference and Competition 2017*, (September):37–43, 2017.

[9] D.C. Van Wijngaarden, E.J.J. Smeur, and B.D.W. Remes. Flight Code Convergence : Fixedwing , Rotorcraft , Hybrid. *IMAV 2021: International Micro Air Vehicle Conference and Competition 2021*, September:21–27, 2021.

[10] E.J.J. Smeur, M. Bronz, and G.C.H.E. de Croon. Incremental control and guidance of hybrid aircraft applied to a tailsitter unmanned air vehicle. *Journal of Guidance, Control, and Dynamics*, 43(2):274–287, 2020.

[11] Open jet facility. https://www.tudelft.nl/lr/organisatie/afdelingen/aerodynamics-wind-energy-flight-performance-and-propulsion/facilities/low-speed-wind-tunnels/open-jet-facility. Accessed: 2022-05-29.

http://www.imavs.org/

Miniaturization and Control of an Unmanned Tiltwing Aircraft

J. Müller [*†], D. Moormann
RWTH Aachen University, Templergraben 55, 52062 Aachen, Germany

http://www.imavs.org/

ABSTRACT

The objective of this paper is to miniaturize and control an unmanned tiltwing aircraft. For geometric miniaturization and mechanical realization, a lightweight design based on carbon fibre tubes, polystyrene rigid foam and 3D-printed parts is used. The attitude controller is adopted from an existing full-scale tiltwing aircraft and follows the control strategy of Incremental Nonlinear Dynamic Inversion (INDI). The effects of the miniaturization, including low mass inertia combined with comparatively large aerodynamic surfaces, require a re-evaluation of INDI-based assumptions. Due to the aircraft's rear-positioned centre of gravity, the modelled control ability of the main motors has to be adapted and requires special consideration in the controller. The integrity of both, construction and controller, is validated in simulations and flight tests.

1 INTRODUCTION

In drone development the trend of miniaturization is applied particularly on concepts based on multi-rotors [1], fixed-wings [2] and flapping-wings [3]. Aircrafts with transformation and vertical take-off and landing (VTOL) ability, such as tiltwing aircrafts, are usually designed for long range outdoor applications [4, 5] and have not yet been subject to miniaturization.

In this paper, the miniaturization process of a tiltwing aircraft based on an existing tiltwing aircraft with a targeted take-off weight of less than $200\,\mathrm{g}$ and a minimum airspeed of $v = 6\,\mathrm{m/s}$ in wing-borne flight is presented. Hereafter, the miniaturized tiltwing aircraft is referred to as Idefix.

Additionally, an attitude controller based on the control strategy of INDI [6] is implemented. Here, the validity of assumptions concerning the negligibility of specific aerodynamic effects is re-evaluated.

The neglect of the distance between thrust vector and centre of gravity is reviewed and the accuracy of the control ability model is investigated accordingly. A validation of both the aircraft construction and the flight controller is conducted through simulation studies and flight tests.

[*]e-mail address: julian.mueller@fsd.rwth-aachen.de
[†]ORCID: 0000-0002-2404-8004

2 MINIATURIZATION OF THE TILTWING AIRCRAFT

Starting point for the miniaturization is the existing tiltwing aircraft Maverix, which was designed at the Institute for Flight System Dynamics of the RWTH Aachen University and has been subject of research since 2014 [7], see Figure 1. With a wingspan of $b = 1.45\,\mathrm{m}$, a wing area of $S = 0.25\,\mathrm{m}^2$ and an empty weight of $m = 1.9\,\mathrm{kg}$, its minimum airspeed in wing-borne flight is $v = 11\,\mathrm{m/s}$.

The goal of the miniaturization is to build the lightest possible aircraft, while retaining all key functions of Maverix. This especially applies to the transformation ability trough a tiltable wing around the wing quarter line in a range of $\alpha_0 < \sigma < 90^\circ$ with α_0 being the angle of incident in wing-borne flight and 90° being the tilt angle for hover flight. The control devices are also kept identical to Maverix, see Figure 1. Less focus is placed on true to scale miniaturization and identical open-loop flight behaviour of both aircrafts.

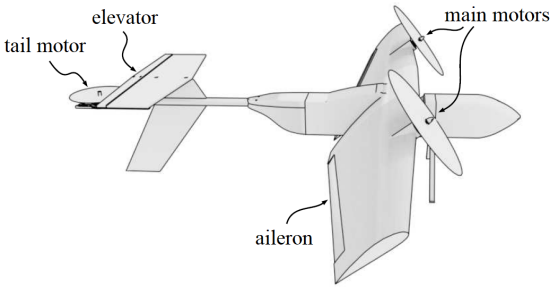


Figure 1: Control devices of Maverix [6]

In addition to requirements for design speed and take-off weight, the following Top Level Aircraft Requirements (TLAR) are set for Idefix:

TLAR		
maximum take-off weight	MTOW	$200\,\mathrm{g}$
minimum speed in wing-borne flight	v_{min}	$6\,\mathrm{m/s}$
minimum flight duration	t_{min}	$5\,\mathrm{min}$
maximum wingspan	b_W	$0.5\,\mathrm{m}$
carrying of a flight computer	m_{fc}	$30\,\mathrm{g}$

Table 1: TLAR for Idefix

For detailed design, Idefix is broken down into following subsystems: propulsion system (1), fuselage (2), wing (3) and tail (4), see Figure 2.

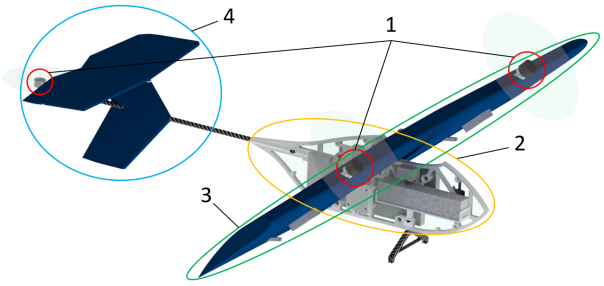


Figure 2: Subsystems of Idefix (CAD)

2.1 Propulsion system design

To ensure sufficient manoeuvrability and provide the capability to carry payload, the combined thrust of the two wing-fixed main motors is set to be at least 1.5 times the aircraft's gravitational force $G_{A/C}$. With a weight of $m = 200\,\mathrm{g}$ this results in a minimum thrust of $F_{M,min} = 1.47\,\mathrm{N}$ per main motor. To increase the efficiency, a motor with low rotation speed and a comparatively large propeller is chosen. A combination of a motor with 4500 revolutions per minute and input voltage (kV) and a $4 \times 2.5\,\mathrm{in}$ propeller, as shown in Figure 3, fulfils these requirements. Powered by a 2S lithium-ion battery with a nominal voltage of $U_N = 7.6\,\mathrm{V}$, this combination delivers a maximum thrust of $F_{M,max} = 1.7\,\mathrm{N}$ with a system mass of $m_M = 7\,\mathrm{g}$.

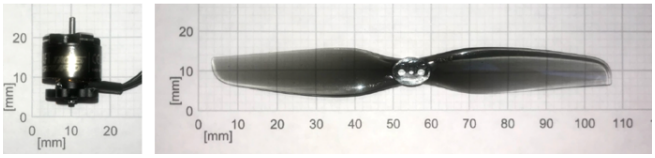


Figure 3: Main motor $(4500\,\mathrm{kV})$ and main propeller $(4 \times 2.5\,\mathrm{in})$

The tail motor is also powered by a 2S lithium-ion battery. It contributes less to the vertical thrust in hover flight and mainly affects the moment equilibrium about the pitch axis. The highest thrust requirement for the tail motor occurs in hover flight configuration, where the absolute value of the main motor and tail motor lever arm ratio becomes minimal with $|\frac{l_T}{l_M}| = 10$. To maintain the pitch moment equilibrium at full main motor thrust, the tail motor needs to deliver a thrust according to Equation 1.

$$F_{T,min} = 2 \cdot F_{M,max} \cdot \left| \frac{l_M}{l_T} \right| = 0.34\,\mathrm{N} \qquad (1)$$

This requirement is met with a combination of a motor with 12000 kV and a $2.5 \times 1\,\mathrm{in}$ propeller, see Figure 4, which yields a thrust of $F_{T,max} = 0.5\,\mathrm{N}$ with a system mass of $m_T = 2.5\,\mathrm{g}$.

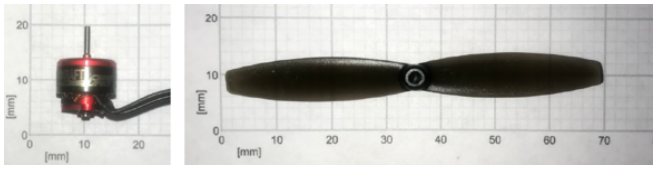


Figure 4: Tail motor (12000 kV) and tail propeller $(2.5 \times 1\,\mathrm{in})$

2.2 Fuselage design

Due to the primarily two-dimensional load on the fuselage consisting of forces in x- and z-direction and moments about the y-axis, the fuselage structure is also chosen to be two-dimensional. To minimise the structural weight and ensure fast reproducibility with a high degree of detail, the structure is 3D-printed in one piece, see Figure 5.

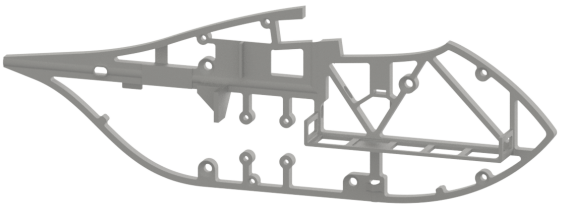


Figure 5: CAD-construction of the fuselage

Besides its function of mechanically connecting all aircraft subsystems, the fuselage also accommodates all main electronic components. Since the available space is limited due to miniaturization, special effort has to be made to ensure electromagnetic compatibility between the electronic components. By splitting the installation space in the symmetry plane via an electromagnetic shielding fleece, the disturbance signal strength of the high-current harness affecting the signal transmission harness can be reduced by up to 100 dB [8, 9], which ensures undisturbed data transmission.

2.3 Wing design

Most conventional design tools like XFLR5 [10] do not consider specific tiltwing effects, such as additional lift produced by higher flow speed at the wing area behind the propeller and tilt of the thrust vector due to the wing's tilt angle. To meet the requirements set in Section 2, an element based simulation developed at the Institute of Flight System Dynamics of the RWTH Aachen University [11], which has been proven to be suitable for the design of tiltwing aircrafts, is used to redesign the wing. For Idefix the same wing profile (NACA 4415) is used as for Maverix, which features both a non-critical stall characteristic and easier lightweight design due to the large profile thickness. The element-based simulation yields, that a wing area of $S_W = 0.05\,\mathrm{m}^2$ is sufficient

http://www.imavs.org/

for a design speed of $v = 6\,\mathrm{m/s}$ and a angle of incident of $\alpha_0 = 8^\circ$ in wing-borne flight. Fully utilising the required maximum wingspan of $b_W = 0.5\,\mathrm{m}$ and simultaneously attempting to achieve an approximately elliptical lift distribution with easy manufacturability and geometric similarity to Maverix results in a wing geometry according to Figure 6 with a mean chord length of $\bar{t} = 0.1\,\mathrm{m}$.

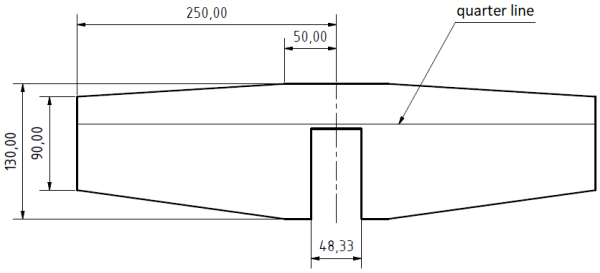


Figure 6: 2D-drawing of the wing [mm]

The lightweight construction of the wing is realised by enforcing a hollow polystyrene structure with a carbon fibre tube. The shielding properties of the carbon fibre can also be taken advantage of by running the data transmission harness inside the tube and the current transmission harness outside, thus resulting in an electromagnetic insulation comparable to the effectivity of the shielding fleece in the fuselage design.

2.4 *Tail and control surface design*

The dimensionless volume coefficients C_h, C_v, C_{ail} and C_{el} are used for scaling the area of the control surfaces of the horizontal stabiliser S_h, the vertical stabiliser S_v, the aileron S_{ail} and the elevator S_{el}, see Equation 2 and 3 [12].

The size and lever arms of Idefix's control surfaces are designed to yield dimensionless volume coefficients identical to Maverix and hence provide sufficient controllability for Idefix. Greater geometrical similarity between both aircrafts is achieved by keeping the aspect ratios of all control surfaces of Idefix as well as the ratio of aircraft width and length identical to those of Maverix.

$$C_h = \frac{S_h \cdot l_h}{S_W \cdot \bar{t}} \;\; ; \;\; C_v = \frac{S_v \cdot l_v}{S_W \cdot b_W} \tag{2}$$

$$C_{ail} = \frac{S_{ail} \cdot l_{ail}}{S_W \cdot b_W} \;\; ; \;\; C_{el} = \frac{S_{el} \cdot l_{el}}{S_W \cdot \bar{t}} \tag{3}$$

The tail structure is kept simple, only consisting of a tail beam made from a carbon fibre tube, which connects both the tail motor and the rear control surfaces with the fuselage, see Figure 7. Analog to the wing, all aerodynamic surfaces are made from polystyrene rigid foam and the insulating effect of the carbon fibre tube is used to minimise electromagnetic interference.

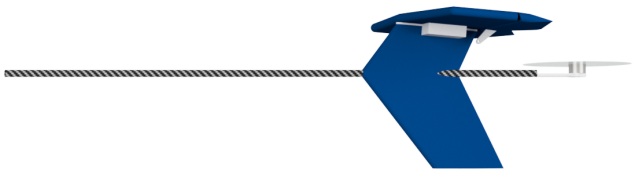


Figure 7: CAD-Construction of the tail

2.5 *Mass properties*

Through consequent lightweight construction, an empty weight of $m = 175\,\mathrm{g}$ can be achieved for the fully assembled Idefix, which is more than 10 times lighter than Maverix. The approximated mass inertia tensor, see Equation 4, is about 100 times lower than that of Maverix. The non-diagonal elements I_{xy}, I_{xz} and I_{zy} as well as changes due to the variable tilt angle of the main wing are negligible and therefore not considered.

$$I = \begin{bmatrix} 884.39 & 0 & 0 \\ 0 & 1456.38 & 0 \\ 0 & 0 & 2160.97 \end{bmatrix} \cdot 10^{-6}\, kgm^2 \tag{4}$$

The approximate coordinates of the centre of gravity are $x_{cg} = -61\,\mathrm{mm}$, $y_{cg} = 0\,\mathrm{mm}$ and $z_{cg} = 19.5\,\mathrm{mm}$, measured from the wing tip in wing-borne flight position in body-fixed coordinates. It is located behind the neutral point and therefore, combined with the low mass inertia, results in higher requirements for the controller compared to Maverix, as is discussed later. In wing-borne flight, the rear-positioned centre of gravity goes along with higher lift demand at the horizontal stabiliser and lower lift demand at the main wing. Thus the required lift can be generated at lower airspeeds.

3 CONTROLLING OF IDEFIX

For the flight controller, the principle of INDI is applied. The theory of INDI is not repeated here and can be found in [6, 13, 14]. By keeping up the central assumption that changes of moments and forces through control devices are much greater than changes of the aerodynamic moments of the aircraft, which in literature is referred to as *time scale separation principle* [15], the control law can be written as follows [6]:

$$u = u_0 + \Delta u = u_0 + B^{-1} \cdot I \cdot (\nu - \dot{\omega}_0). \tag{5}$$

Here, u is the control device command at a certain time step, u_0 the control device command from the previous time step and B^{-1} the pseudo-inverse [16] of the control ability matrix B, which is not directly invertible due to the aircraft's over-actuation [13]. The quantities ν and $\dot{\omega}_0$ represent the pseudo-control input for the INDI controller, which is a vector of commanded angular accelerations and the angular acceleration from the previous time step, respectively. The

http://www.imavs.org/

closed loop INDI controller following Equation 5 is shown in Figure 8 [6]. Here, $\hat{A}$ is the model of the real control device dynamic A and $\hat{B}$ the model of the real control ability B.

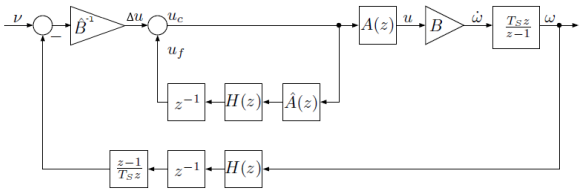


Figure 8: Closed loop INDI Controller [6]

Even though the controller is robust against errors in control ability modelling, the control quality benefits from an accurate modelling of both the control abilities and the control device dynamics. For a more detailed description of the INDI Controller, see [6].

3.1 Control ability

Whereas the control ability for asymmetric thrust, tail motor, ailerons and elevator can be adopted from Maverix [13], the neglect of the pitch effect due to symmetric thrust has to be re-evaluated. This neglect is valid for Maverix because the distance between thrust vector F_M and centre of gravity is small. Thus, the resultant pitch ability B_M, which is equivalent to the change of moment through change of the main motor throttle signal $\frac{\partial M_M}{\partial \delta_M}$, is significantly lower than the tail motor pitch ability B_T. For Idefix, the centre of gravity is located further behind the tilt axis. The pitch ability of the main motors with the kinematic correlation shown in Figure 9 can be described with Equation 6, 7 and 8.

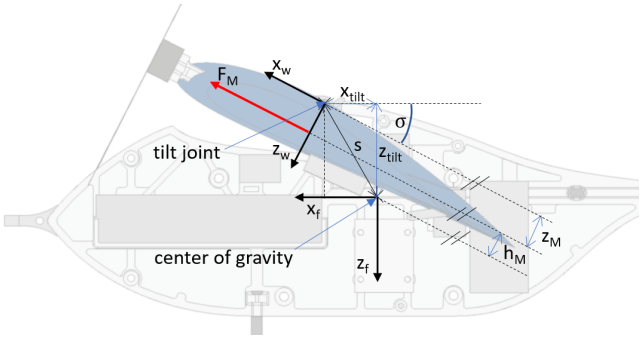


Figure 9: Kinematic correlation of the pitch ability generated by the main motors

$$B_M = \frac{\partial M_M}{\partial \delta_M} = \frac{\partial F_M}{\partial \delta_M} \cdot h_M(\sigma) \tag{6}$$

$$h_M = \sin(90\,^\circ - \arctan\left(\left|\frac{x_{tilt}}{z_{tilt}}\right|\right) - \sigma) \cdot s - z_M \tag{7}$$

$$s = \sqrt{x_{tilt}^2 + z_{tilt}^2} \tag{8}$$

Here, the index w represents the wing-fixed coordinate system [13] with origin in the tilt joint and f the aircraft-fixed coordinate system with origin in the center of gravity. The parameters x_{tilt} and z_{tilt} describe the horizontal and vertical distance between the origins of those coordinate systems, z_M is the orthogonal distance between the thrust vector and the origin of the wing-fixed coordinate system and h_M the effective lever arm between the thrust vector and the origin of the aircraft-fixed coordinate system.

This effect depends on the current tilt angle σ and is maximal in hover flight, where its pitch ability amounts to approximately 33 % of the pitch ability of the tail motor. It contributes much more to the pitch moment equilibrium than it does for Maverix and thus it has to be investigated if the pitch ability needs to be considered in the modelled control ability.

3.2 Consideration of the additional control ability

An alternative approach for calculating the control device commands is replacing the pseudo-inverse from Equation 5 with an optimization algorithm (allocator), which is also used for Maverix [6]. It allows for the prioritization of individual control devices as well as their limitations and hence offers more adjustment options, resulting in more suitable control device commands for the aircraft.

The simplest way to consider the additional pitch control ability is adding it to the existing control ability matrix B, yielding an expanded control ability matrix B_{exp}, see Figure 10.

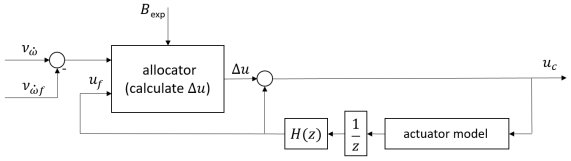


Figure 10: Calculation of the changes of the control device commands by an allocator [6]

Another approach is not considering the additional pitch ability as a control ability but as a predictable disturbing effect with the disturbing ability $B_{M,dist}$. In this case, the original control ability matrix B remains unaltered and the disturbing effect caused by the disturbing ability $B_{M,dist}$ is equalised by adding a tail motor command $\Delta u_{T,add}$ to the previously calculated tail motor command, with Δu_T being the change of the main motor command and B_T being the control ability of the tail motor, see Figure 11 and Equation 9.

$$\Delta u_{T,add} = \Delta u_M \cdot \frac{B_{M,dist}}{B_T} \tag{9}$$

Even though this case avoids using the allocator to find an optimal solution for the control device commands by considering all control abilities, it serves as an alternative approach for compensating the main motor pitch ability. Its effect on the pitch behaviour is analysed in the next section.

http://www.imavs.org/

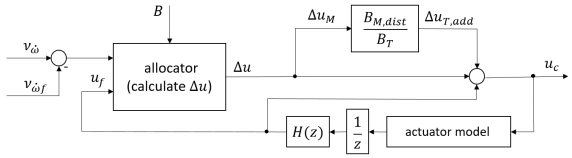


Figure 11: Consideration of the main motor pitch ability as a disturbing effect

4 Validation

The validation of the controller is conducted in both simulation and free flights. An element-based approach implemented in MATLAB/Simulink is used for the simulation [11]. Despite the aim for indoor applicability, the flight tests are executed on an airfield for safety reasons.

4.1 Simulation

For the simulative validation of the attitude controller, the step responses to commanded changes in the reference variables ϕ, θ and $\dot{\psi}$ are investigated. Figure 12 shows the results in hover flight, with a rise time of $1\,\mathrm{s} < t_r < 2\,\mathrm{s}$ and a settling time of $t_s < 3\,\mathrm{s}$ in all axis. The response behaviour is overall stable and comparable to that of Maverix [13].

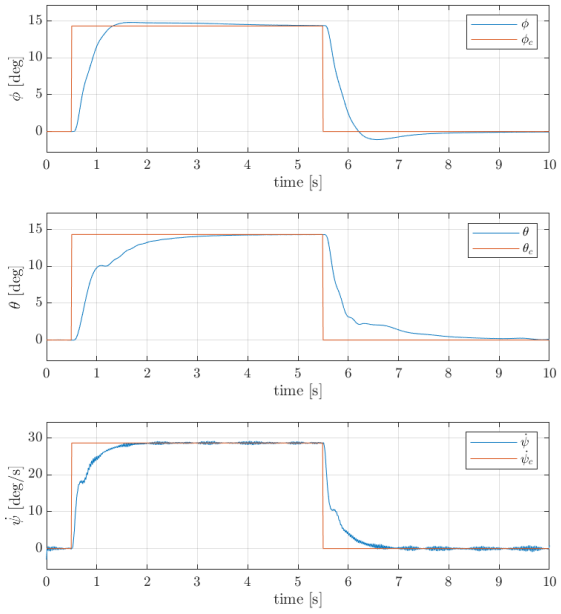


Figure 12: Step response of the attitude controller in hover flight (simulation)

In wing-borne flight, the step response shows similar results in roll and pitch axis, with a slightly higher rise time and an overshoot of less than $2\,^\circ$, see Figure 13. The response in the yaw axis only yields an initial rise of $50\,\%$ and does

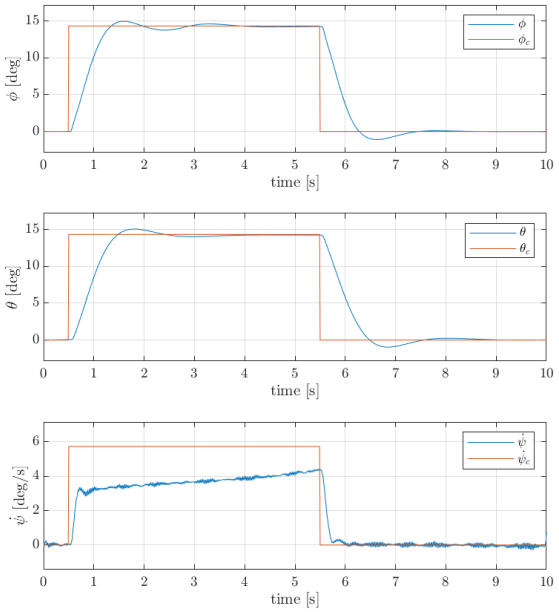


Figure 13: Step response of the attitude controller in wing-borne flight $v \approx 7\,\mathrm{m/s}$ (simulation)

not reach the commanded step within five seconds, whereas for the return of the commanded yaw rate to $0\,^\circ/\mathrm{s}$ a fast response behaviour with a rise time of $t_r = 0.25\,\mathrm{s}$ is observed. This lack in response also occurs in the angular acceleration $\dot{r}$, suggesting that a yawing moment exists, that counteracts the commanded angular acceleration.

Since the step in $\dot{\Psi}$ is isolated and does not include a command in ϕ for a coordinated turn, a slip angle β builds up, see Figure 14.

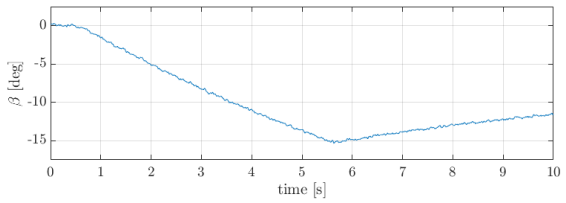


Figure 14: Build-up of a slip angle through a commanded yaw motion (simulation)

By approximating the vertical stabiliser with a finite flat plate of area S_v, aspect ratio Λ_v and a mean lever arm l_v to the aircraft's centre of gravity, the temporal change of yaw moment can be expressed by Equation 10 [13].

$$\frac{\partial N(\dot{\beta})}{\partial t} = \frac{\rho}{2} v^2 \cdot S_v \cdot 2\pi \cdot \frac{\Lambda_v}{\Lambda_v + 2} \cdot \dot{\beta} \cdot l_v \qquad (10)$$

http://www.imavs.org/

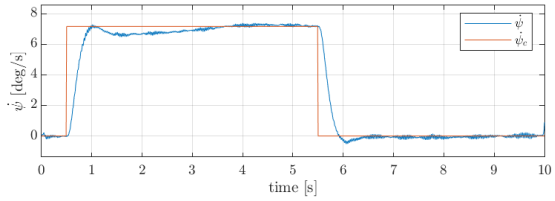


Figure 15: Command response of the yaw rate with adjusted mass properties (simulation)

An evaluation of this Equation shows that the change of yaw moment caused by the change of the slip angle is of the same order as the controller-commanded change of torque for overcoming the aircraft's mass inertia. If the mass properties of Idefix are adjusted to match Maverix's ratio of mass inertia and aerodynamic damping, the response behaviour changes, as shown in Figure 15. This response behaviour fits the command significantly better, which supports the idea that the building-up damping effect of the vertical stabiliser is responsible for the observed lack in response. While neglecting this effect does not impact the command response for Maverix, it does for Idefix. To consider this effect in the controller, the slip-angle has to be determined continuously, which is accompanied by more measurement effort. However, flight states with high slip-angles are rarely required and Idefix's yaw response behaviour while reducing the slip angle is sufficiently fast. Therefore, a special consideration of this effect is not implemented.

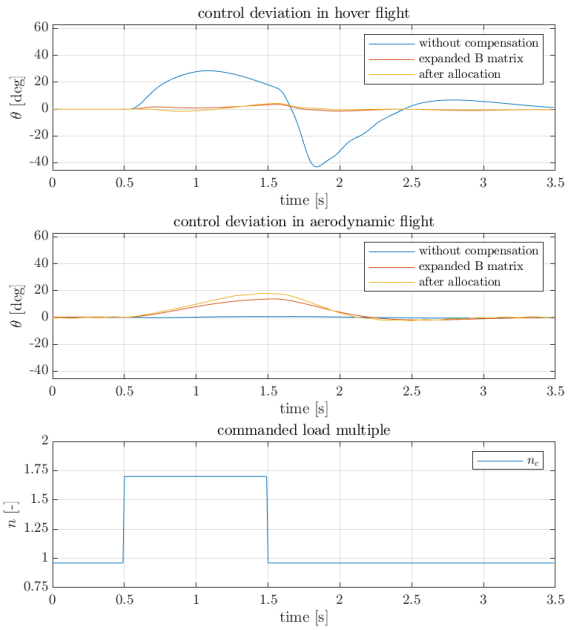


Figure 16: Pitch angle deviation (simulation)

The compenzation of the thrust-induced pitch moment introduced in Section 3.1 and 3.2 is investigated through the response of the pitch angle to commanded load multiples. Figure 16 shows a significant improvement in the control error in hover flight compared to that of the uncompensated case with a reduction of deviation from $40\,^\circ$ to less than $5\,^\circ$. Yet in wing-borne flight, a worsening of the control deviation is observed.

This behaviour can be ascribed to the interaction of the slip stream behind the main motors with the main wing. The inflow downstream the propeller plane is higher than the aircraft's airspeed and thus generates more lift. Figure 17 shows the kinematic correlation between both the additional lift F_{add} and the lever arm h_{add} and the tilt angle σ as well as the geometric parameters of the aircraft x_{tilt} and z_{tilt}. This interaction results in an additional pitch moment $M_{A,add}$, see Equation 11,12 and 13.

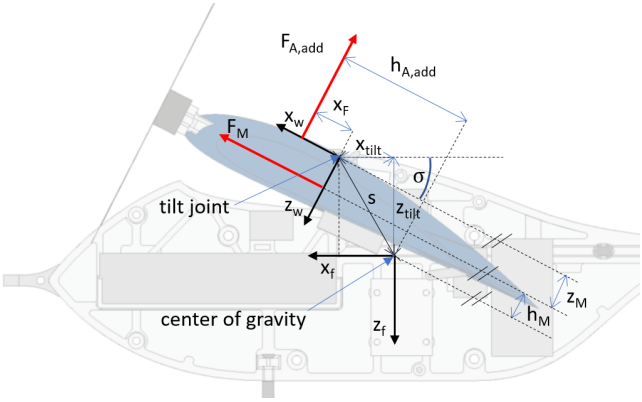


Figure 17: Kinematic correlation of the additional lift induced by the main motors

$$M_{A,add} = F_{A,add} \cdot h_{A,add} \tag{11}$$

$$h_{A,add} = \cos(90^\circ - \arctan\left(\left|\frac{x_{tilt}}{z_{tilt}}\right|\right) - \sigma) \cdot s + x_F \tag{12}$$

$$s = \sqrt{x_{tilt}^2 + z_{tilt}^2} \tag{13}$$

Particularly noteworthy is the moment's change of sign at a tilt angle of $\sigma = 50\,^\circ$. This change means that the pitch moment induced by motor thrust is amplified in a range of $50\,^\circ < \sigma < 90\,^\circ$ and reduced in a range of $0\,^\circ < \sigma < 50\,^\circ$, with a complete cancellation at $\sigma = 50\,^\circ$. Even though it is easy to determine the direction of this moment, the quantification is challenging due to uncertainties in the location of the centre of pressure and consequently the lever arm $h_{A,add}$ as well as the amount of additional force $F_{A,add}$.

To minimise the interaction of these effects without conducting additional wind tunnel experiments, the pitch compenzation is only applied in the tilt angle range of $50\,^\circ < \sigma < 90\,^\circ$, where both effects have the same sign.

http://www.imavs.org/

4.2 *Free Flight*

Free flights with Idefix show that in hover flight the attitude angle accurately follows the commanded changes and rarely exceeds a deviation of more than 2 °, see Figure 18.

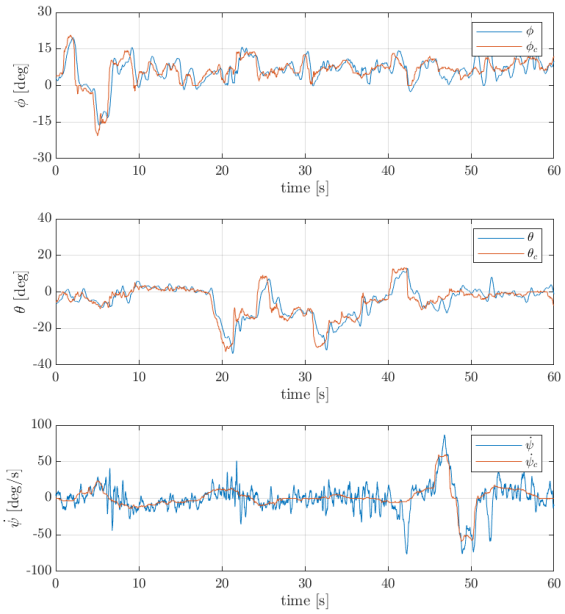


Figure 18: Command response of the attitude controller in hover flight (free flight)

Also in wing-borne flight, a well fitting command response is observed, see Figure 19. This includes yaw rates of up to $\dot{\Psi} = 40\,°/s$ during coordinated turns, which confirms the legitimacy of the previously described negligence of the yaw moment caused by the build-up of a slip angle. Only during turns a control deviation of up to 5 ° is observed, which can be explained by cross flow due to imperfect turns as well as harsh wind conditions on this particular flight.

For the pitch compensation, only flight tests in hover flight are conducted due to the argumentation in Section 4.1, see Figure 20 - 22. These include attitude control without the main motor pitch compensation, compensation with an extended control ability matrix and compensation with consideration as a disturbing effect.

The changes in the load factor are manually commanded and therefore do not represent real steps. To allow comparability with the simulation results, the load factor signals used in the free flights are also run through the simulation used in Chapter 4.1. It is observed that in all cases the real deviation in pitch angle exceeds the simulated deviation, which may be due to the differences between the real aircraft and the simulation. Even though the simulation shows similar results for both compensations, the smallest deviation in the pitch angle during free flight is yielded with compensation through

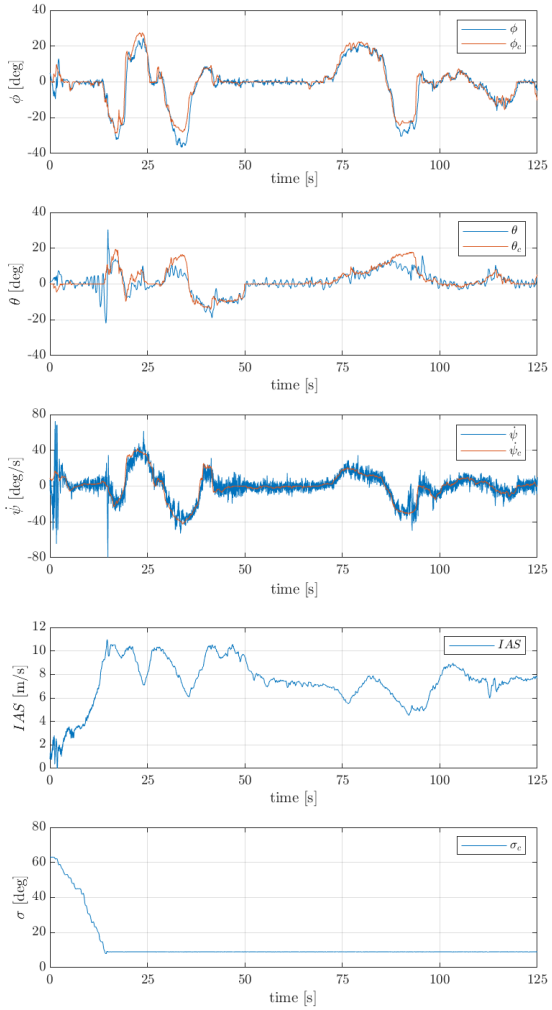


Figure 19: Command response of the attitude controller in wing-borne flight (free flight)

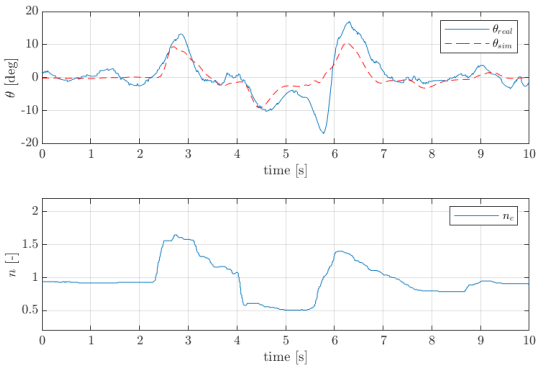


Figure 20: Change of pitch angle without pitch compensation (free flight)

http://www.imavs.org/

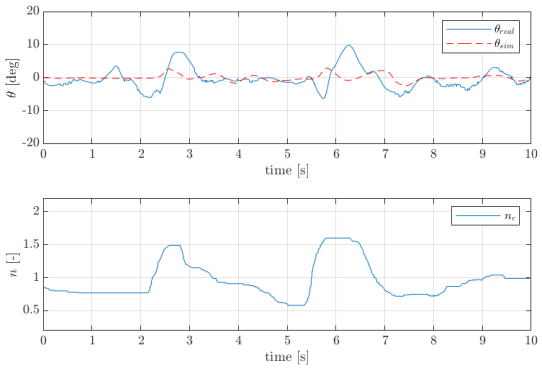


Figure 21: Compensation with extended control ability matrix (free flight)

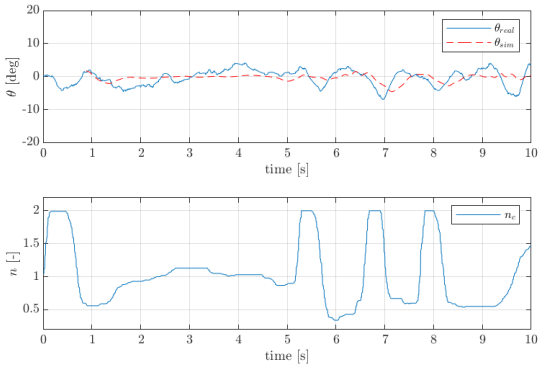


Figure 22: Compensation through consideration as disturbing effect (free flight)

consideration as a disturbing effect. However, as described in Section 4.1, this avoids finding an optimum solution with the allocator and therefore should be used as a reference for further optimization of the allocator.

4.3 Flight performance

All requirements for flight performance set in chapter 2 could be fulfilled for Idefix, as shown in the take-off ready configuration in Figure 23. These include a maximum flight duration of 5 min in hover flight and up to 10 min in wing-borne flight. In stationary horizontal wing-borne flight, an airspeed of $v = 6\,\mathrm{m/s}$ could be achieved at a pitch angle of $\theta = 0\,°$, see Figure 24. Though this already satisfies the requirement, even lower airspeeds in horizontal flight can be achieved at higher angles of attack.



Figure 23: Take-off ready Idefix

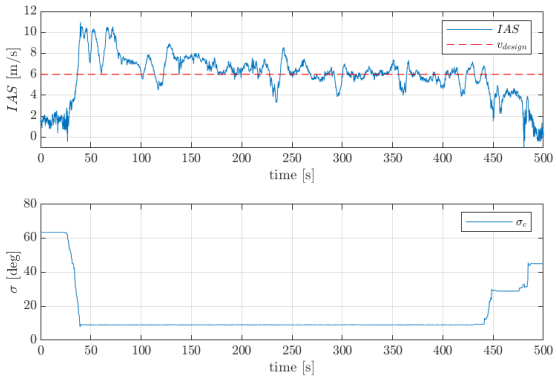


Figure 24: Air speed during free flight

5 Conclusion

In this work the miniaturization of a tiltwing aircraft was conducted. Through consequent lightweight design, the wingspan could be reduced by 66 % and the take-off weight by more than 90 % compared to Maverix, resulting in a take-off weight of $m = 175\,\mathrm{g}$. Special challenges resulting from the miniaturization, such as electromagnetic interference between electronic components due to the small installation space, were identified and solved constructional.

An attitude controller based on the concept of INDI was implemented with focus on expanding the modelling of the main motor's control ability and its implementation in the controller. This was necessary due to the aircraft's rear-positioned centre of gravity and thus the main motors' noticeable pitch ability. In slip flight, damping aerodynamic effects of the vertical stabiliser, which were negligible for Maverix, were identified and quantified for Idefix.

Both the aircraft and the controller were validated in simulations and flight tests and showed stable flight behaviour in both hover and wing-borne flight. With respect to the TLARs, all requirements, including a minimal flight duration of $t = 5\,\mathrm{min}$ and a minimal flight speed of $v = 6\,\mathrm{m/s}$, were fulfilled.

Future work will concentrate on finding a solution for the aircraft's position determination, which complies with the requirements for fully automated flight in GPS-denied areas and improving flight behaviour, especially during the transition from hover to wing-borne flight.

http://www.imavs.org/

REFERENCES

[1] Domenica Costantino, Maria Giuseppa Angelini, and Gabriele Vozza. The engineering and assembly of a low cost UAV. In *2015 IEEE Metrology for Aerospace (MetroAeroSpace)*, pages 351–355, 2015.

[2] Mostafa Hassanalian, Hamed Khaki, and Mehrdad Khosravi. A new method for design of fixed wing micro air vehicle. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 229(5):837–850, 2015.

[3] G. C.H.E. De Croon, M. A. Groen, C. De Wagter, B. Remes, R. Ruijsink, and B. W. Van Oudheusden. Design, aerodynamics and autonomy of the DelFly. *Bioinspiration and Biomimetics*, 7(2), 2012.

[4] FSD Homepage - RWTH Aachen University `https://www.fsd.rwth-aachen.de/cms/fsd/Forschung/~mhauu/Fluggeraete/?lidx=1`. Accessed: 2022-08-01.

[5] Paul M. Rothhaar, Patrick C. Murphy, Barton J. Bacon, Irene M. Gregory, Jared A. Grauer, Ronald C. Busan, and Mark A. Croom. NASA langley distributed propulsion VTOL tilt-wing aircraft testing, modeling, simulation, control, and flight test development. *AIAA AVIATION 2014 -14th AIAA Aviation Technology, Integration, and Operations Conference*, pages 1–14, 2014.

[6] Fabian Binz. Robust, Fault-Tolerant Control of Aircraft with Hovering Capability, 2020.

[7] P Hartmann, M Schütt, and D Moormann. Vollständigen Flugbereich Eines Kippflügelflugzeuges. *Deutscher Luft- und Raumfahrtkongress 2014 DocumentID: 340131*, pages 1–9, 2014.

[8] YSHIELD GmbH & Co KG. Electromagnetic Shielding Fleece `https://www.yshield.com/en/artikel_pdf/item/416/variation/1389/`. Accessed: 2022-08-01.

[9] YSHIELD GmbH & Co KG. Electromagnetic Shielding Fleece - Expert report of RF transmission attenuation `https://pdf.yshield.com/pdf/db/EXPERTREPORT-YSHIELD-HNV100.pdf`. Accessed: 2022-08-01.

[10] A Deperrois. About XFLR5 calculations and experimental measurements, 2009.

[11] Philipp Hartmann. Predictive Flight Path Control for Tilt-Wing Aircraft, 2017.

[12] Ingmar Buchweitz. Flugzeugentwurf am Beispiel eines ausgeführten Flugzeugs, 2002.

[13] F. Binz, T. Islam, and D. Moormann. Attitude control of tiltwing aircraft using a wing-fixed coordinate system and incremental nonlinear dynamic inversion. *International Journal of Micro Air Vehicles*, 11, 2019.

[14] Erik-jan Van Kampen. Incremental backstepping for robust nonlinear flight control Incremental Backstepping for Robust Nonlinear Flight Control, 2016.

[15] Ewoud J.J. Smeur, Qiping Chu, and Guido C.H.E. De Croon. Adaptive incremental nonlinear dynamic inversion for attitude control of micro air vehicles. *Journal of Guidance, Control, and Dynamics*, 39(3):450–461, 2016.

[16] João Carlos Alves Barata and Mahir Saleh Hussein. The Moore-Penrose Pseudoinverse: A Tutorial Review of the Theory. *Brazilian Journal of Physics*, 42(1-2):146–165, 2012.

http://www.imavs.org/

Attitude control for a tiltwing aircraft under tail actuator failures

D. Schatten* and D. Moormann
RWTH Aachen University, Templergraben 55, 52062 Aachen, Germany

http://www.imavs.org/

ABSTRACT

In this paper we present a concept for attitude control of a tilt wing aircraft under tail rotor failures. During hover flight, the exampled tilt wing aircraft flies as a conventional tri-copter with two main motors and one auxiliary tail rotor. In case of an tail rotor failure the body swings beneath the main wing and the aircraft stabilizes as a flying wing. Using a body-fixed coordinate system, angular accelerations are controlled by a robust attitude controller on the basis of Nonlinear Dynamic Inversion (INDI). Combined with a height controller and an LQR based angular acceleration controller, a controlled flight and descend can be achieved.

1 INTRODUCTION

The combination of vertical take-off and landing (VTOL) capabilities with the ability to fly as conventional airplanes, make tilt wing aircraft a versatile design. While at cruising speed, the tilt wing aircraft uses its main wing to generate lift. During VTOL the aircraft rotates its main wing upwards and the main engines thrust is used to climb and hover. To stabilize and enable pitching maneuvers during hover, an auxiliary tail rotor is employed. During this VTOL phase, engine failures lead to uncontrolled descend if not compensated by an emergency maneuver or redundant engine designs [1] [2] [3] Redundant engine design concepts, like including more engines than needed for thrust generation always have the negative downside that as the overall system mass increases and therefore payload or flight time is reduced. To address this problem, emergency maneuvers can be employed. While in fast forward flight, classical one engine out procedures can be applied [1] . However, engine failures in hover flight pose a critical flight situation due to the lack of thrust generation and limited recovery altitude. In this paper we present a control structure which enables an unmanned tilt wing aircraft to safely control its attitude while maintaining its altitude during tail rotor failure. Taking into consideration that the motor thrust in combination with the tilt angle affects pitch momentum as well as the longitudinal acceleration, a prioritization for the stability around pitch axis while maintaining altitude is favorable.

*Email address(es): schatten@fsd.rwth-aachen.de

2 MODEL

The aircraft which is being stabilized is depicted in figure 1. This tilt wing has a span of 1.5 m with an average weight of around 2 kg. While in normal hovering mode, the main rotors induce momentum around the roll axis, while the auxiliary rotor is used for pitch control. Yaw control is enabled by aileron deflections. During transition flight , the main wing is rotated along the body fixed y-axis referred to as σ. In cruising configuration, the tilt wing aircraft is actuated like a conventional aircraft with aileron, elevator and thrust control. Differential trust is used as yaw-control.

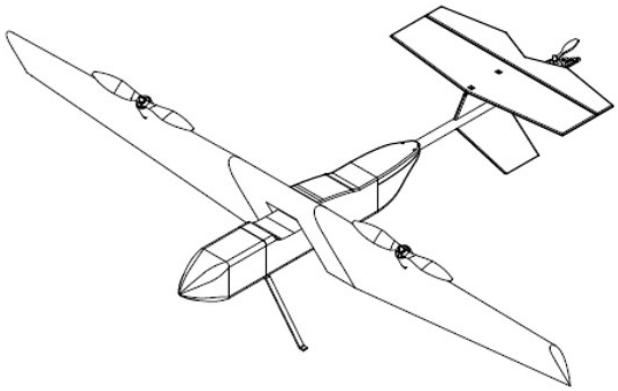


Figure 1: Exampled tilt wing in hover mode

	hover flight	emergency mode
asym. thrust δ_{asym}	roll	roll
tail rotor δ_{aux}	pitch	(-)
aileron ξ	yaw	yaw
elevator η	(-)	(-)
tiltrotor σ	(-)	pitch

Table 1: Actuators in hover flight and emergency mode

During tail rotor failure the attitude controller stabilizes the aircraft around the trimm state where the thrust vector points to earth centered -z_g. While in hover mode the motion of the aircraft can be described by the following set of equations:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} \dot{u}' \\ \dot{v}' \\ \dot{w}' \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} \dot{u}' \\ \dot{v}' \\ \dot{w}' \end{bmatrix} = \begin{bmatrix} \frac{F_t \cdot cos(\sigma)}{m} - g \cdot sin(\Phi) - F_{aero_x} \\ g \cdot cos(\Theta) \cdot sin(\Phi) - F_{aero_y} \\ \frac{-F_t \cdot sin(\sigma)}{m} - g \cdot cos\Phi \cdot cos(\Theta) - F_{aero_z} \end{bmatrix} \tag{2}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = I^{-1} \cdot \left[\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \left(I \cdot \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right) \right] \tag{3}$$

$$\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} M_{mot_x} + F_{ail} \cdot y_{cog} \cdot sin(\sigma) + M_{aero_x} \\ M_{mot_y} + M_{servo} + M_{aero_y} \\ M_{mot_z} + F_{ail} \cdot y_{cog} \cdot cos(\sigma) + M_{aero_z} \end{bmatrix} \tag{4}$$

$$\begin{bmatrix} M_{mot_x} \\ M_{mot_y} \\ M_{mot_z} \end{bmatrix} = \begin{bmatrix} F_{dt} \cdot y_{cog} \cdot cos(\sigma) \\ F_t \cdot z_{cog} \cdot cos(\sigma) + F_t \cdot x_{cog} \cdot sin(\sigma) \\ F_{dt} \cdot y_{cog} \cdot sin(\sigma) \end{bmatrix} \tag{5}$$

With $[\dot{u}\ \dot{v}\ \dot{w}]^T$ being the body fixed translation accelerations, $[\dot{p}\ \dot{q}\ \dot{r}]^T$ the body fixed rotational accelerations, the mass m of the aircraft, $I = [I_{xx}\ I_{yy}\ I_{zz}]^T$ inertia around the main body axes, F_t the main thrust of the front motors, F_{dt} the differential thrust between the left and right front motor, F_{ail} the force induced by deflection of the ailerons, $[F_{a_x}\ F_{a_y}\ F_{a_z}]$ aerodynamic forces, $[M_{aero_x}\ M_{aero_y}\ M_{aero_z}]$ aerodynamic moments, $[M_{mot_x}\ M_{mot_y}\ M_{mot_z}]$ motor induced moments, $[x_{cog}\ y_{cog}\ z_{cog}]^T$ being the pivot point of the wing-body and span-wise position of motors and ailerons.

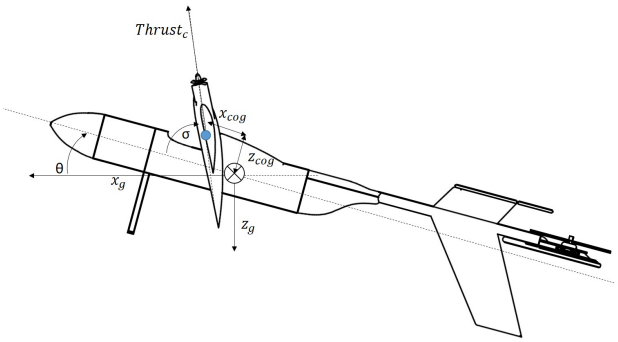


Figure 2: Model of the tilt wing aircraft during tail actuator failure

The aerodynamic effects are computed by breaking down the aircraft structure into aerodynamic plates and adding up the forces and momentum according to [4] . While the aerodynamic effects relative to the air are accounted for in M_{aero}, M_{servo} models the added moment due to the torque applied to the wing and therefore the equivalent reaction moment on the center of gravity:

$$M_{servo} = M_{rot} + I_{yy_{wing}} \cdot \ddot{\sigma} \tag{6}$$

By using a point mass model of the wing, we approximate the inertia $I_{yy_{wing}}$ of the wing and therefore account for the effects of angular acceleration $\ddot{\sigma}$.

M_{rot} accounts for the aerodynamic effects due to angular wing tilt velocity $\dot{\sigma}$. By separating the wing into aerodynamic plates above and below the wing pivot point, we are able to derive equations for the aerodynamic forces acting on the wing.

$$M_{rot} = F_1 \cdot \frac{t_1}{2} + F_2 \cdot \frac{t_2}{2} \tag{7}$$

By modeling our aerodynamic elements as plates with span wise length $b = 1.5\ m$, root wise length t and drag coefficient $c_d = 1.2$ and assuming the main drag forces acting on the center of each plate, we can calculate each force as

$$F = c_d \cdot \frac{\rho}{2} \cdot \dot{\sigma}.^2 \cdot t^3 * b \tag{8}$$

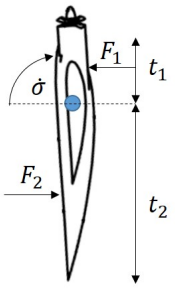


Figure 3: Modeling of the aerodynamic effects due to a change in σ

2.1 Trimm state in emergency mode

Modeling the aircraft as a point mass and neglecting aerodynamic forces due to small rotational velocity, the system can be described around the pitch axes as a pendulum with longitudinal movement. While the thrust generated by the two main rotors is needed to counteract gravitational forces, momentum is generated around the pitch axis. During tail rotor recovery, the body swings beneath the wing pivot point in a rear centered cog while the main thrust vector is used for stabilization and height control. The aircraft which is being

http://www.imavs.org/

stabilized has a rear centered center of gravity (cog). While a cog below the wing pivot would lead to a neutral moment and therefore enable the aircraft to hover vertical with respect to the body angle θ, a rear centered cog was chosen to increase the margin for a shift of cog due to added mass in front of the designed cog. The main advantage of a rear centered cog in contrast to a forward centered cog is a lift contribution of the tail rotor in normal mode. Figure 4 depicts the time frame recovery phase of the aircraft under tail rotor failure.

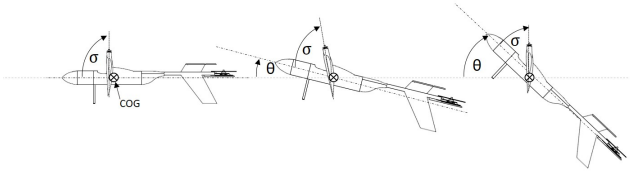


Figure 4: Movement of the aircraft during recovery

3 CONTROL STRATEGY

In this chapter we will discuss the overall control strategy and give more insight to the separate controllers and control allocation. To stabilize the unmanned aircraft under tail-rotor failures the following control structure is employed:

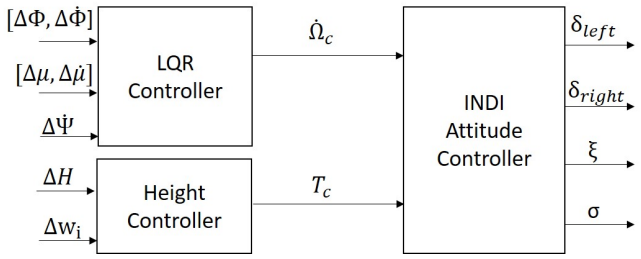


Figure 5: Control structure

The controller depicted in this paper is part of an cascaded controller loop. In the inner controller loop an INDI attitude controller is employed. The INDI attitude controller's task is to stabilizes the aircraft in normal and emergency mode by using body fixed angular acceleration $\dot{\Omega}_c$ and Thrust command T_c as input and actuator commands described in table 1 as output. To achieve this dual purpose task an fault detection system and emergency moding system is employed to redesign the INDI controller by adjusting the actuator effectivity matrix B. During tail rotor-failure, a PD height controller is employed to generate a thrust command T_c according to the height and inertial velocity deviations ΔH and Δw_i. Angular acceleration commands $\dot{\Omega}_c$ are generated by separate LQR based acceleration controllers which use the deviations of roll angle $\Delta\Phi$ with rate feedback, thrust angle $\Delta\mu$ with rate feedback and yaw velocity $\Delta\dot{\Psi}$ respectfully. Further details regarding the controller parts are depicted in the following subsections.

3.1 Incremental Nonlinear Dynamic Inversion

The concept of Incremental Nonlinear Dynamic Inversion (INDI) has already been discussed in several papers [5] [6] and will not be explained in detail. The basic concept of INDI is to invert the plant dynamics and therefore shape the closed-loop dynamics as a series of integral states. A simple derivation of the INDI control law is as follows.

$$\dot{x} = f(x) + g(x) \cdot u \tag{9}$$

Equation 9 represents the formulation of a multi input multi output (mimo) system. f represents the effects of aerodynamic and flight dynamic characteristics affecting the aircraft. g models the effects of input vectors on the system. By inverting equation 9 we gain:

$$u = g(x)^{-1} \cdot (\dot{x} - f(x)) \tag{10}$$

This control law enables the transformation from the current system state to an desired system state by controlling the state input u. The downside for this simple approach is the needed modeling effort around the mostly unknown aerodynamic effects in $f(x)$. To reduce the modeling effort, [5] [7] propose an incremental dynamic approach by employing a taylor series around the state vector x and input vector u. This approach reduces the impact of model mismatches and therefore aerodynamic effects are compensated.

The resulting governing control law for a rotational incremental nonlinear dynamic inversion is depicted in the following equation:

$$\Delta u = B^{-1} \cdot I \cdot (\dot{\Omega}_c - \dot{\Omega}) \tag{11}$$

where B is the inverted actuator effectivity matrix with their associated dynamics, I the moment of inertia and $\delta(\dot{\Omega})$ the difference between commanded and measured angular accelerations. The governing control law therefore computes incremental steps on the actuators command to transfer the system from the current rotational state to an desired rotational state.

3.2 Angular acceleration controller

In this chapter we will describe the development and implementation of an LQR-based angular acceleration controller. This angular acceleration controller uses the angle deviation of a controlled trim state as input. The basic principle of LQR control is to minimize a cost function designed with weighting factors either by prioritizing state variables or actuator usage. The cost-function in time discrete finite horizon formulation is depicted in equation 12.

http://www.imavs.org/

$$J = x_N^T \cdot Q \cdot x_N + \sum_{k=0}^{N-1} (x_k^T \cdot Q \cdot x_k + u_k^T \cdot R \cdot u_k + 2 \cdot x_x^T \cdot N \cdot u_k) \tag{12}$$

Q and R are positive, symmetric weighting factors prioritizing either state error variables x or actuator input u. By minimizing the quadratic cost function depicted in equation 12, a feedback control law can be developed to minimize state error:

$$u = -K_{LQR} \cdot x \tag{13}$$

By implementing a time-discrete LQR-controller, we can model reaction times concerning actuator dynamics and computational time. Study's on our flight controller and flight data have shown, that a time delay of $\delta t = 7 \cdot controller_{freq.}$ occurs. Due to the principle of INDI inverting the plant dynamics, we are able to develop a linear model of our plant by modeling our INDI attitude controller as a transfer function of first order with a time constant of $T_{INDI} = 0.1$ and a time delay of $t_{delay} = 0.035s$. To account for the time delay we need to extend our classical model of linear state space with additional time delayed state variables and extended matrixes as depicted as follows:

$$\begin{aligned} \dot{x}_{n+1} &= A \cdot x_n + B \cdot u_n \\ y_n &= C \cdot x_n + D \cdot u_n \end{aligned} \tag{14}$$

Depicted in equation 14 is the linear state space model around our body fixed pitch axis consisting of an extended state vector $x_n = [\mu_n \; \dot{\mu_n} \; \ddot{\mu_n} \cdots \ddot{\mu}_{n-7}]^T$,terestrial based indi input $u = (\dddot{\mu_c})$ extended system matrix $A \in R^{10x10}$ extended input matrix $B \in R^{10x1}$, extended output matrix $C \in R^{10x10}$ and feedthrough matrix $D = 0$. While μ and $\dot{\mu}$ are measurable state variables, states regarding $[\ddot{\mu}_n, ...\ddot{\mu}_{n-7}]^T$ need to be estimated. Assuming our INDI model representation is correct, we are able to construct an linear observer model which uses $\ddot{\mu_c}$ as input and estimates $x_{n_{observer}} = [\dddot{\mu_n} \; \ddot{\mu}_{n-1} \; ... \; \ddot{\mu}_{n-7}]^T$. By using the transformation matrix M_{bi} from terrestrial to the body fixed frame, we are able calculate an angular pitch acceleration $\dot{q_c}$ for the INDI attitude controller

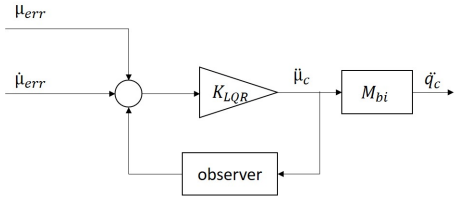


Figure 6: LQR Controller

3.3 Height controller

Task of the height controller is at first to maintain altitude during tail actuator failure recovery. Once a steady state attitude is reached, a controlled descend is initiated. To achieve this goal, a force F_z is generated by a PD controller which acts contrary to the gravitational forces along the z_g axis.

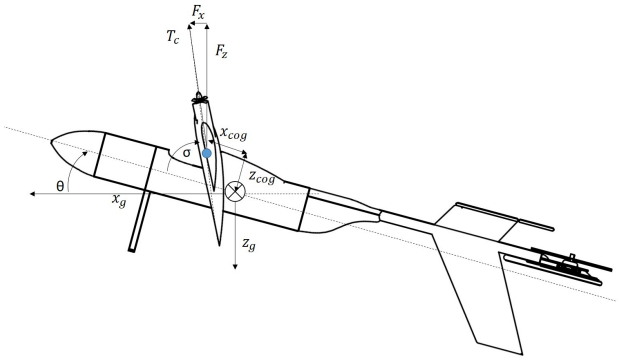


Figure 7: Height control

By considering equation 15 we can calculate a Thrust command T_c depending on the pitch θ and tilt angle σ of the aircraft.

$$T_c = \frac{F_z}{sin(\sigma + \theta)} \tag{15}$$

3.4 Actuator modeling

The control law depicted in equation 11 uses the inverted model of actuators to form an inner control loop. The actuator models consist of static and dynamic properties and describe the change in momentum due to changes in the actuator deflection. As mentioned in [8], the actuator modeling affects the closed-control loop dynamics and can be modeled as a compromise between controller dynamics and modeling effort. In this paper we only adress extensions to the actuator modeling as described by [6] regarding the pitch authority.

As depicted in equation 3 the influence of a deviation of σ or T lead to a pitching moment. In this chapter we will describe the contribution of a tilting motion of the main wing and thrust to the pitching moment. By partial differentiating equation 3 with respect to σ, we can express the influence of σ:

$$\frac{\delta M}{\delta \sigma} = x_{COG} \cdot cos(\sigma_0) \cdot F_{main} - F_{main} \cdot z_{COG} \cdot sin(\sigma_0) \tag{16}$$

where σ_0 denote the tilt angle in the previous controller time step and F_{main} is the current thrust approximated by the following motor model:

$$F_t = (V_{Bat} \cdot K_{Bat} - K_{offset}) \cdot \delta^2 \tag{17}$$

http://www.imavs.org/

where V_{Bat} is the measured supply voltage, $\delta \in [0...1]$ the normalized throttle signal, K_{Bat} and K_{offset} modeling parameters derived from experimental data. By adding up the modeled thrust for the left and right motor we are able to calculate the main thrust acting on the tilt wing aircraft in near stationary flight.

$$F_{main} = F_{t_{left}} + F_{t_{right}} \tag{18}$$

As for the dynamic model of our servo, we employed a non-linear model consisting of a time delay of $t_{delay} = 0.035s$,and rate limit corresponding to the data from our test bench [8].Assuming the servo velocity acts instantaneous with the maximum velocity we are able to model $\dot{\sigma}$ and $\ddot{\sigma}$. The influence of a thrust change on the pitch momentum can be modeled similar to the modeling of the wing tilt effects. By differentiating equation 3 with respect to F_t we can derive the following equation:

$$\frac{\delta M}{\delta_\delta} = (x_{COG} \cdot sin(\sigma_0) - \cdot z_{COG} \cdot cos(\sigma_0)) \cdot \frac{\delta_{F_t}}{\delta_\delta} \tag{19}$$

where σ_0 is the previous wing angle and $\frac{\delta_{F_t}}{\delta_\delta}$ the derivative of equation 15 with respect to δ.

4 Simulation and Results

To validate our controller design presented in the previous chapters, we will conduct simulation studies with regard to the area of application for our emergency mode and controller performance.

4.1 Controller performance during attitude recovery

To evaluate our controller and get a first understanding of the motion of the modeled aircraft, we simulated an tail rotor failure by deactivating tail- thrust in our aircraft plant at a given time $t_{failure}$. Due to the lack of tail thrust, emergency mode was activated and the attitude controller stabilizes the aircraft. Initial setup is a hover flight with $[u_g\ v_g\ w_g] = [0\ 0\ 0]\frac{m}{s}$, $[\phi\ \mu\ \psi] = [0\ 90\ 0]^\circ$ and no wind speed $V_w = 0\ \frac{m}{s}$.

Figure 8 shows the translation movement and angles during hover and tail rotor failure recovery. Until reaching the failure time $t_{failure} = 7s$ the modeled aircraft remains in steady hover flight where the inertial velocities remain near 0. By deactivating the tail thrust, θ starts to increase while the wing-tilt angle σ decreases at maximum speed to stabilize the aircraft around its trim state. At the time $t = 12s$ a steady flight is achieved.

Figure 9 shows the attitude LQR controller performance during tail rotor recovery. In general the performance of the LQR controller with regard to thrust angle μ compared to yaw or roll commands can be considered sluggish. This is due to a relative large difference between θ_{start} and θ_{trimm} and a compromise between attitude angles and angular velocities. Studies regarding over prioritization of angle commands have shown an increasing overshoot of the commanded trim angle.

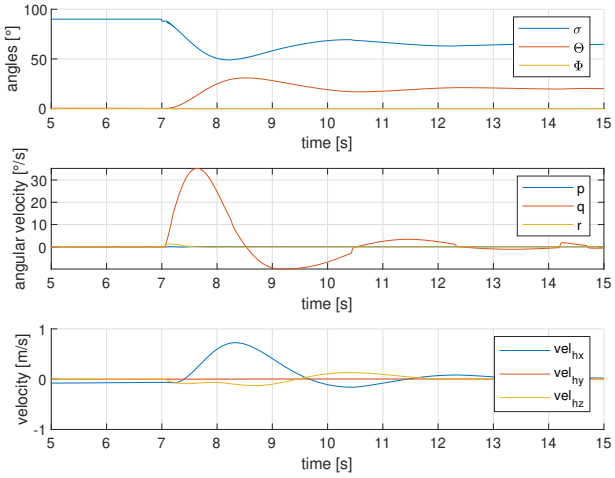


Figure 8: Motion of the aircraft during tail rotor recovery

This is due to the increased induced momentum generated by thrust. On the other side a over prioritization of pitch damping resulted in an increasing height loss and longitudinal acceleration.

By using our closed loop dynamic model of our tilt servo we are able to obtain expected accelerations regarding our pitch axis. By filtering the commanded angular accelerations, we are able to asses the performance of our INDI inner control loop. Figure 10 shows exemplary the INDI time series with our expected $\dot{q}_c$ and measured $\dot{q}$. We can see that around $t_{failure} = 7s$ our INDI model mismatches our measured angular accelerations. The main cause for the model mismatch are aerodynamic moments which are not modeled in the INDI controller. Studies have shown, that during recovery, the longitudinal aerodynamic velocity buildup leads to an pitch moment due to aerodynamic forces acting above the cog.

Figure 11 shows the deviation of a commanded height including horizontal velocities during tail rotor recovery. The horizontal coordinate system composes the terrestrial fixed velocities vel_g transferred to the horizontal system vel_h by angle ψ. The overall performance of the controller regarding a height loss during tail rotor recovery is very satisfactory. As mentioned before, our developed emergency mode is a compromise between height loss and longitudinal velocity buildup with a prioritization on reduction of uncontrolled height loss. While the height loss is minimal $\Delta H = 0.25\ m$, a longitudinal velocity buildup $u_h = 0.9\ m/s$ and following reduction can be observed.

To investigate the controller performance based on the initial attitude, simulations with deviating starting attitudes have been performed. Figure 12 shows exemplary the attitude performance of the controller. In this setup stationary flights with deviating roll angles ϕ have been performed. At

http://www.imavs.org/

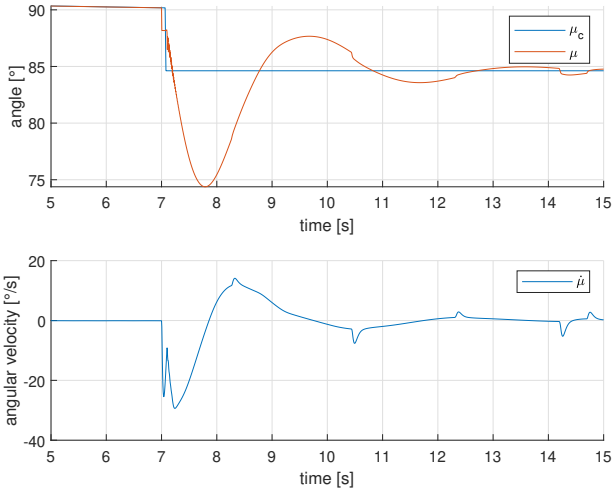


Figure 9: Attitude control performance during tail rotor failure recovery

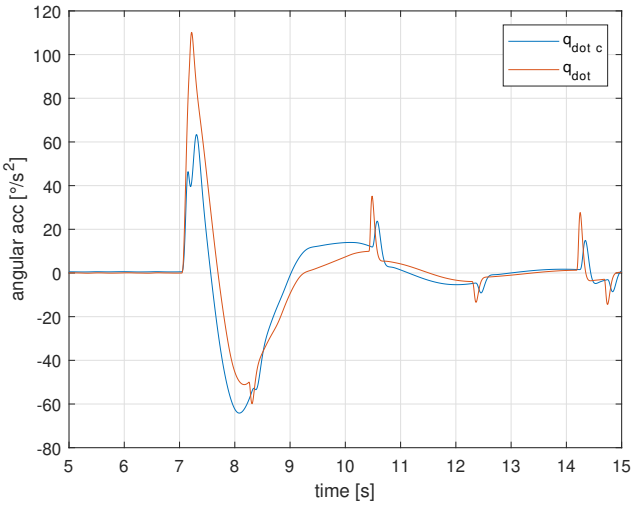


Figure 10: INDI controller performance

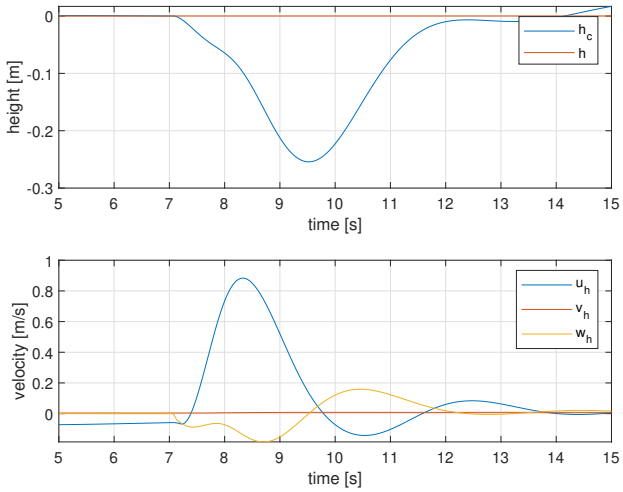


Figure 11: Height loss during emergency recovery

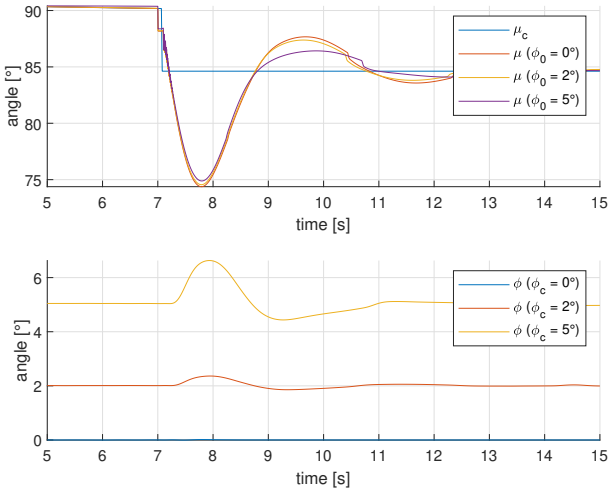


Figure 12: Attitude controller performance based on initial attitude

$t_{failure} = 7s$ emergency mode was activated and a tail rotor recovery was performed. While increasing initial roll angles lead to a larger deviation in ϕ, μ is largely unaffected. Roll angles above $\phi > 7$ have shown an insufficient heading control due to actuator saturation. This can be traced back to aerodynamic forces acting on the vertical stabilizer resulting from lateral velocity $v_{yh} = 6.5\ m/s$ which is considered an unrealistic flight state. Studies regarding initial $\dot{\psi}$ have shown similar results.

4.2 *Maneuverability in stabilized mode*

To evaluate the performance and boundary's of the emergency controller we applied step and ramp responses after the recovery from a tail rotor failure around the pitch and roll axis. As we can see in figure 13, the overall performance of the controller during a step response is acceptable. While the pitch controller can still be considered sluggish although not as sluggish during tail rotor compensation, roll is considered good.

By analyzing a ramp response of our controller we are able to evaluate a steady state reaction of our system regarding actuator behavior and saturation effects. Figure 14 shows a near steady state reaction where the pitch angle μ is reduced from stationary state. With a decreasing thrust angle μ, wing angle σ is decreased as well while the body pitch angle θ increases. The reason for this is still under investigation but

http://www.imavs.org/

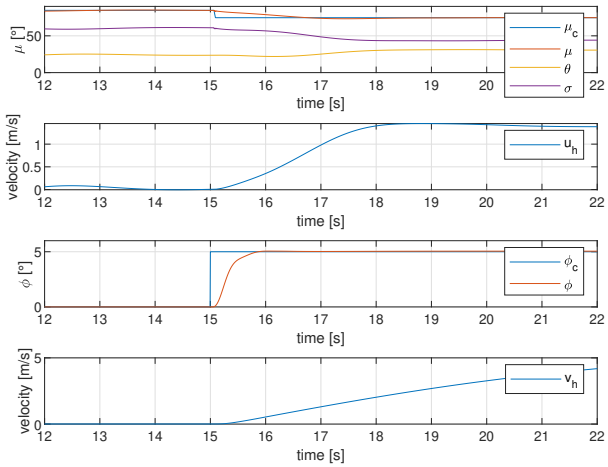


Figure 13: Step responses around roll and pitch

is most likely attributed to the overall aerodynamic effects acting on the aircraft. With further decreasing σ, θ starts to decrease as well, which is due to the overall influence of the motor thrust acting on the cog. At around $t = 47s$ σ starts to oscillate with a high frequency. Studies regarding the INDI indicate that modeling errors occur and need to be further investigated.

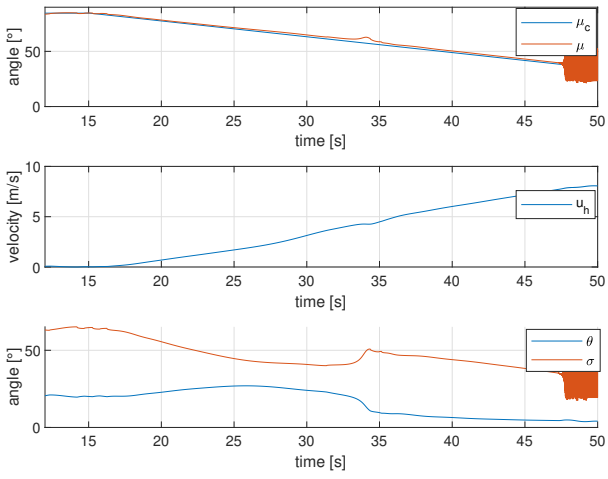


Figure 14: Ramp response around pitch axis.

Generally we are able to control our aircraft in emergency mode around all axis. By influencing our thrust vector, we are able to induce force and build up horizontal velocities. By analyzing our height controller it became obvious, that a steady descent and climb is achievable while still being stabilized.

4.3 *Controller performance under wind effects*

In this chapter we will analyze the robustness of our proposed control structure during the transition phase from normal to emergency mode. We will put our emphasis on the effects of wind gust and wind speeds ranging from expected wind speeds of $u_{hw} = [0...10]\ \frac{m}{s}$. To evaluate our controller performance with regard to wind effects, we employed a wind model with averaged wind and gustiness. The task of the aircraft is to remain stationary in terms of terrestrial velocities. Therefore we expanded our control structure by implementing a PI controller with input Δu_h being the deviation of inertial longitudinal velocity and output the thrust pitch angle μ for longitudinal motion control.

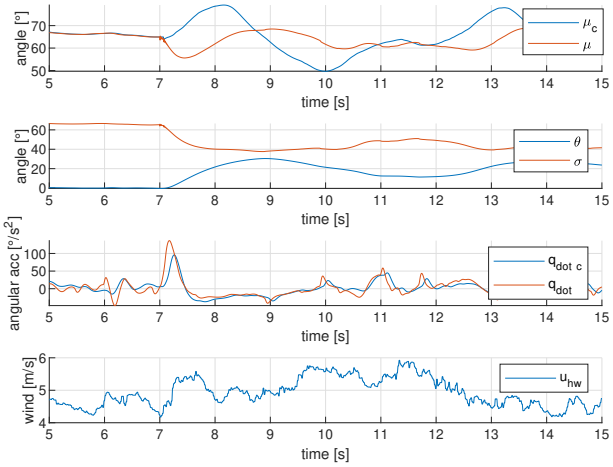


Figure 15: Controller performance under wind effects

Figure 15 displays exemplary the performance of the controller under wind effects at speeds $u_{hw} = 5\ \frac{m}{s}$. Although the controller is rather sluggish around its pitch axis we are able to maintain stationary in the context of terrestrial velocities until $u_{hw} = -u_a = 7\frac{m}{s}$. When exceeding $u_{hw} > 7$ oscillations similar to figure 14 occurred. Studies related to 4.1 and 4.2 under wind effects have shown similar performance.

5 CONCLUSION AND FUTURE WORK

In this work we presented an alternative approach to increase safety of unmanned aerial vehicles by employing an emergency mode to compensate for tail rotor failures rather than redundant design concepts. The proposed control structure transforms an tilt wing aircraft with 3 motors into an basic flying wing. We employed a robust INDI controller combined with an LQR attitude controller and height control. Simulation studies have shown, that our aircraft was able to stabilize after tail rotor failure under different wind and starting attitude conditions. While height loss was minimal, we encountered longitudinal accelerations, which lead

http://www.imavs.org/

to forward and backward movement. This effect was considered adequate as a primary object of the proposed control structure was to reduce the risk of ground damage and a appropriate horizontal airspace around our aircraft is available. We were able to induce force for a controlled flight and therefore enable an controlled flight after tail rotor failure. The overall controller performance was acceptable while the pitch authority remains sluggish. Future work will concentrate on the improvement of pitch control. Particularly the compromise between angular damping and angular commands will be further analyzed. Furthermore test flights in an controlled flight space are performed.

REFERENCES

[1] Hamidreza Jafarnejadsani Donglei Sun Florian Holzapfel Xiang Fang, Neng Wan and Naira Hovakimyan. Emergency landing trajectory optimization for fixed-wing uav under engine failure. 2019.

[2] G. Narvydas R. Maskeliūnas R. Damaševičius B. AL-Madani, M. Svirskis. Design of fully automatic drone parachute system with temperature compensation mechanism for civilian and military applications. *Journal of advanced Transportation*, 2018.

[3] R. Isermann. *Fault- Diagnosis Systems*. Springer Science and Business Media, LLC, Heidelberg, GER, 2006.

[4] P. Hartmann. Vorausschauende flugbahnregelung fuer kippfluegelflugzeuge. 2017.

[5] J.A. Mulder S. Sieberling, Q.PChu. Robust flight control using incremental nonlinear dynamic inversion and angular acceleration prediction. *Journal of Guidance, Control, and Dynamics*, 33(6), 2010.

[6] F. Binz T.Islam, D. Moormann. Attitude control of tiltwing aircraft using a wing-fixed coordinate system and incremental nonlinear dynamic inversion. In *IEEE International Micro air vehicle conference and competition (IMAV)*, 2018.

[7] J. A. Mulder R. R. da Cosat, . P. Chu. Reentry flight controller design using nonlinear dynamic inversion. *Journal of Spacecraft and Rockets*, 40(1), 2003.

[8] F. Binz D. Moormann. Actuator modeling for attitude control using incremental nonlinear dynamic inversion. In *IEEE International Micro air vehicle conference and competition (IMAV)*, 2019.

http://www.imavs.org/

Robust flight control using Hybrid Incremental Nonlinear Dynamic Inversion Control for Tiltrotor UAV

Hoijo Jeaong, Jinyoung Suk*, and Seungkeun Kim
Chungnam National Univeristy, 99 Daehak-ro, Yuseong-gu, Daejeon 34134, Republic of Korea

http://www.imavs.org/

ABSTRACT

This paper uses a complementary filter to design a hybrid incremental nonlinear dynamic inversion controller as a model-based and sensor-based angular acceleration estimation method. The dynamic modeling is performed on the aerodynamic data of the tiltrotor UAV and the propulsion system. By acquiring the dynamic characteristic information of the system, the cross-frequency of the complementary filter is applied. This paper checks the control effectiveness of the hybrid incremental nonlinear dynamic inversion. The control performance against the doublet attitude command is compared for a proportional-integral-differential controller, nonlinear dynamic inversion, and incremental nonlinear dynamic inversion.

1 INTRODUCTION

In recent years, Unmanned Aerial Vehicles (UAVs) have been widely used for civilian and military purposes, such as traffic monitoring, delivery missions, and geographic surveying. They can reduce operating costs by replacing human-crewed aircraft exposed to repetitive missions or dangerous environments [1, 2]. UAVs may require robust control by disturbance depending on the mission environment. Moreover, it may be designed as a nonlinear, highly coupling, uncertain, time-varying system depending on the form of the UAV. Typical control methods have become challenging to meet the system's good performance. Therefore, a control method used for flight control using angular acceleration by differentiating the angular velocity measured in the gyroscope has been proposed [3]. After the control concept using angular acceleration for fighter VAAC came out, it is possible to improve the system's robustness by feedback angular acceleration, such as incremental nonlinear dynamic inversion (INDI) [4, 5], the filter of angular acceleration with noise [6].

There are three main advantages of applying the angular acceleration feedback to the control system.

1. Acceleration or angular acceleration has a close relationship with the force and moment applied to the system according to Newton's law. The effect of vibration or disturbance applied to the system is related to the angular acceleration, which can be used to eliminate model uncertainty and disturbance of the system.
2. INDI is simplified for the information required by the controller in the affine equation of state. It reduces the system's dependence and is robust against modeling errors and uncertainties.
3. Since there is no model dependency compared to nonlinear dynamic inversion (NDI), there is no need for complex gain scheduling. There is no heavy workload for designing flight control laws.

One of the challenges in INDI is how to obtain accurate angular acceleration signals [4]. Accurate angular acceleration is key to reinforcing the robustness of INDI. In general, angular acceleration is calculated numerically by differentiating the angular velocity measured by the gyroscope [7, 8]. At this time, noise increases, and a time delay occurs by the filter, reducing the stability of the flight control system (FCS). Many researchers have done much research on these issues. In Reference [9], the methods for measuring angular acceleration are divided into two methods: direct and indirect. The indirect method is to estimate the angular acceleration using the velocity, and the primary purpose is to attenuate the angular acceleration noise. In addition, studies are using predictive filters [10] and Kalman filters [11]. The above method has many parameters that need to be set, and there is a limit to reflect the system characteristics. In addition, a low-pass filter is usually designed to reduce the influence of noise, but it has a significant disadvantage in terms of time delay.

Motivated by the discussions mentioned above, in this paper, accurate angular acceleration is obtained in a relatively simple way using a complementary filter. In the following, we summarize the significant contributions of this paper:

- Numerical modeling and trim analysis for a tiltrotor UAV.
- Application of cross-frequency of complimentary filter based on dynamic characteristic information obtained in fixed-wing flight mode
- The observer is designed a hybrid incremental nonlinear dynamic inversion (HINDI) controller using model-based and sensor-based angular acceleration signals.
- Comparison of attitude tracking performance of proportional-integral-differential controller (PID),

*Email address(es): jsuk@cnu.ac.kr

NDI, and INDI controllers for HINDI performance comparison.

This paper is structured as follows: In Section 2, numerical modeling of the tiltrotor UAV is performed through aerodynamic data and propulsion system modeling. In Section 3, the method of obtaining angular acceleration is divided into sensor-based INDI and model-based INDI. Moreover, an accurate angular acceleration observer is designed using the complementary filter and model-based and sensor-based angular acceleration information. Section 4 compares the performance of PID, NDI, INDI, and HINDI to compare attitude tracking performance. Finally, Section 5 concludes this paper.

2 NUMERICAL MODELING

2.1 Aerodynamic Modeling

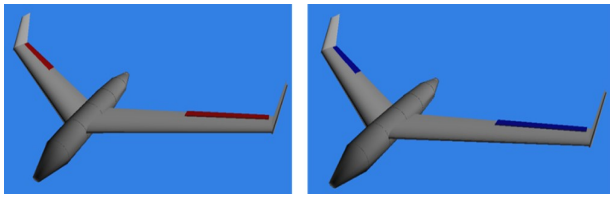


Figure 1: DATCOM Longitudinal (Left), Lateral (Right)

External forces and moments are essential factors to ensure the modeling reliability of the platform to be modeled. The factors of external force and moment acting on each axis include aerodynamic effect, rotor thrust, gravity, and aerodynamic force. The aerodynamic coefficient is generally calculated theoretically or obtained by an experimental approach through a CFD technique or a wind tunnel test. In this study, static/dynamic non-dimensional aerodynamic coefficients are calculated using Digital DATCOM software [12]. The DATCOM can calculate static stability and control surface deflection characteristics and includes an option to calculate the trim at subsonic Mach number.

Equations 1-7 show stability derivatives and force and moment coefficients of the tiltrotor UAV used in this study. Here, ${\delta_{ev}}^R$ and ${\delta_{ev}}^L$, mean the deflection angles of the left and right elevons and $x_{C.G}$ and $x_{A.C}$ mean the positions of the center of gravity and the center of aerodynamics based on the body coordinate system, respectively. If the left and right elevons are deflected in the same direction and magnitude, the sum of the rolling moments due to the $C_{L_{\delta_{ev}}}$ becomes zero. Conversely, if the same magnitude deflects the left and right ailerons in opposite directions, the pitching moment due to the $C_{M_{\delta_{ev}}}$ becomes zero [13].

$$\begin{aligned} C_L =& C_{L_0} + C_{L_\alpha}(V_T,\alpha)\cdot\alpha + \frac{1}{2}C_{L_{{\delta_{ev}}^R}}(V_T,{\delta_{ev}}^R)\cdot\Delta{\delta_{ev}}^R \\ &+ \frac{1}{2}C_{L_{{\delta_{ev}}^L}}(V_T,{\delta_{ev}}^L)\cdot\Delta{\delta_{ev}}^L + \frac{\bar{c}}{2V_T}C_{L_q}(V_T,\alpha)\cdot q \end{aligned} \tag{1}$$

$$\begin{aligned} C_D =& C_{D_0} + \frac{1}{2}C_{D_{{\delta_{ev}}^R}}(V_T,{\delta_{ev}}^R)\cdot\Delta{\delta_{ev}}^R \\ &+ \frac{1}{2}C_{D_{{\delta_{ev}}^L}}(V_T,{\delta_{ev}}^L)\cdot\Delta{\delta_{ev}}^L \end{aligned} \tag{2}$$

$$C_Y = C_{Y_\beta}\cdot\beta + \frac{b}{2V_T}(C_{Y_p}(V_T,\alpha)\cdot p + C_{Y_r}(V_T,\alpha)\cdot r) \tag{3}$$

$$\begin{aligned} C_l =& C_{l_\beta}(V_T,\alpha)\cdot\beta + \frac{1}{2}C_{l_{{\delta_{ev}}^R}}(V_T,{\delta_{ev}}^R)\cdot\Delta{\delta_{ev}}^R \\ &- \frac{1}{2}C_{l_{{\delta_{ev}}^L}}(V_T,{\delta_{ev}}^L)\cdot\Delta{\delta_{ev}}^L \\ &+ \frac{b}{2V_T}(C_{l_p}(V_T,\alpha)\cdot p + C_{l_r}(V_T,\alpha)\cdot r) \end{aligned} \tag{4}$$

$$\begin{aligned} C_M =& C_{M_0} + C_{M_\alpha}(V_T,\alpha)\cdot\alpha + \frac{1}{2}C_{M_{{\delta_{ev}}^R}}(V_T,{\delta_{ev}}^R) \\ &\cdot\Delta{\delta_{ev}}^R + \frac{1}{2}C_{M_{{\delta_{ev}}^L}}(V_T,{\delta_{ev}}^L)\cdot\Delta{\delta_{ev}}^L + \\ &\frac{\bar{c}}{2V_T}C_{M_q}(V_T,\alpha)\cdot q + A \end{aligned} \tag{5}$$

$$C_N = C_{N_\beta}\cdot\beta + \frac{b}{2V_T}(C_{N_p}(V_T,\alpha)\cdot p + C_{N_r}(V_T,\alpha)\cdot r) - B \tag{6}$$

where

$$\begin{aligned} A &= (x_{C.G} - x_{A.C})({C_L}^{Total}\cos\alpha + {C_D}^{Total}\sin\alpha) \\ B &= (x_{C.G} - x_{A.C})\frac{\bar{c}}{b}C_Y^{Total}\cos\alpha. \end{aligned} \tag{7}$$

2.2 Dynamic modeling

The dynamic modeling of the tiltrotor UAV is derived by forces and moments acting on the vehicle. Each force and moment consider the components of the rotor, the elevons, the blend-wing. This paper assumes constant mass and mass distribution during flight. The configuration of the tiltrotor UAV is shown in Figure 2. X_E, Y_E and Z_E axes of the inertial coordinate system are defined as a right-handed Cartesian coordinate with North, East, and Downward directions, abbreviated as the NED coordinate. The dynamics can be represented as

$$\dot{x}(t) = f(x(t),u(t)) \tag{8}$$

where

$$\begin{aligned} x &= [\ u \quad v \quad w \quad p \quad q \quad r \quad \phi \quad \theta \quad \psi \quad]^T, \\ u &= [\ RPM_1 \quad RPM_2 \quad RPM_3 \quad RPM_4 \quad \delta_{ev}^R \quad \delta_{ev}^L \quad \xi \ ], \end{aligned} \tag{9}$$

http://www.imavs.org/

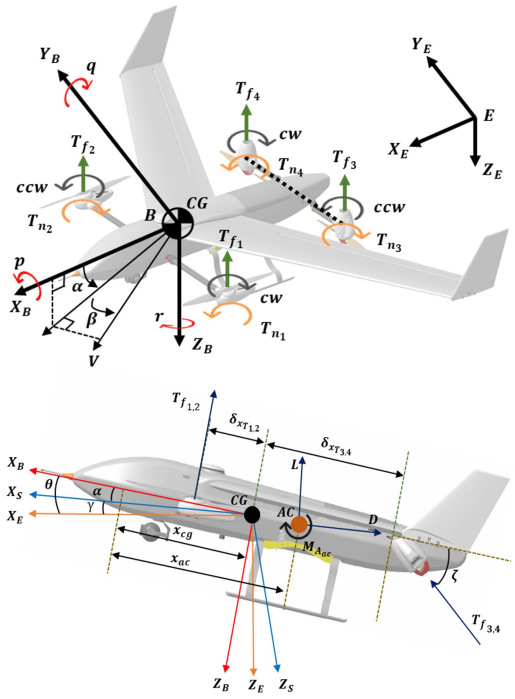


Figure 2: Coordinate System of Tiltrotor UAV

, and $u, v, \& w$ and $p, q, \& r$ are the projections of velocity and angular velocity onto the X_B, Y_B, and Z_B axes of the body-axis system. Euler angles are presented by ϕ, θ, and ψ. Six degree-of-freedom nonlinear equations of motion are derived by reflecting total force and moment as

$$\begin{aligned}
\dot{u} &= rv - qw - g\sin\theta + F_x/m \\
\dot{v} &= pw - ru + g\sin\phi\cos\theta + F_y/m \\
\dot{w} &= qu - pv + g\cos\phi\cos\theta + F_z/m \\
\dot{p} &= (L + qr(I_y - I_z))/I_x \\
\dot{q} &= (M + rq(I_z - I_x))/I_y \\
\dot{r} &= (N + pq(I_x - I_y))/I_z
\end{aligned} \tag{10}$$

Here, the force and moment in each axis are represented as:

$$\begin{aligned}
\bar{F}_B &= \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \bar{F}_{B_T} + \bar{F}_{B_A} \\
&= \begin{bmatrix} T_{f_3}\cos\zeta + T_{f_4}\cos\zeta \\ 0 \\ T_{f_1} + T_{f_2} + T_{f_3}\sin\zeta + T_{f_4}\sin\zeta \end{bmatrix} + \\
&\quad \begin{bmatrix} QSC_L\sin\alpha - QSC_D\cos\alpha \\ QSC_Y \\ -QSC_L\cos\alpha - QSC_D\sin\alpha \end{bmatrix}
\end{aligned} \tag{11}$$

$$\bar{M}_B = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = \bar{M}_{B_T} + \bar{M}_{B_g} + \bar{M}_{B_A} \tag{12}$$

where $Q = 0.5\rho V^2$, and

$$\bar{M}_{B_T} = \begin{bmatrix} T_{f_1}L_2 - T_{f_2}L_2 + T_{f_3}L_4\sin\zeta \\ -T_{f_4}L_4\sin\zeta \\ T_{f_1}L_1 + T_{f_2}L_1 - T_{f_3}L_3\sin\zeta \\ -T_{f_4}L_3\sin\zeta \\ -T_{q_1}L_2 + T_{q_2}L_2 + T_{q_3}L_4 - T_{q_4}L_4 \\ +T_{f_3}\cos\zeta\sin\delta_g\sqrt{L_3^2 + L_4^2} \\ -T_{f_4}\cos\zeta\sin\delta_g\sqrt{L_3^2 + L_4^2} \end{bmatrix} \tag{13}$$

$$\bar{M}_{B_g} = \begin{bmatrix} -J_r q(-\Omega_1 + \Omega_2 \\ +\Omega_3\sin\zeta - \Omega_4\sin\zeta) \\ -J_r r(\Omega_3 - \Omega_4)\cos\zeta \\ +J_r p(-\Omega_1 + \Omega_2 + \Omega_3\sin\zeta - \Omega_4\sin\zeta) \\ J_r q(\Omega_3 - \Omega_4)\cos\zeta \end{bmatrix} \tag{14}$$

$$\begin{aligned}
\bar{M}_{B_A} &= \begin{bmatrix} QSbC_l \\ QScC_m \\ QSbC_n \end{bmatrix} + \\
&\begin{bmatrix} -(x_{CG_Z} - x_{AC_Z})\bar{F}_{B_{A_Y}} \\ (x_{CG_Z} - x_{AC_Z})\bar{F}_{B_{A_X}} - (x_{CG_Z} - x_{AC_Z})\bar{F}_{B_{A_Z}} \\ (x_{CG_Z} - x_{AC_Z})\bar{F}_{B_{A_Y}} \end{bmatrix}.
\end{aligned} \tag{15}$$

Here, $\bar{F}_B$ and $\bar{M}_B$ are forces and moments based on the body coordinate system, $T_{n_{1\sim4}}$ means the rotor's thrust, and ζ means the tilt angle of the rear rotor [14]. $L_{1\sim4}$ are the horizontal and vertical distances from the rotor to the center of gravity. The torque component resulting from the rotor consists of each pair's thrust difference and the gyroscopic effect [15].

3 Control Design

3.1 Sensor-Based INDI

The INDI is a concept expressed in the incremental form to improve the robustness of the closed-loop system and reduce the dependence on system information compared to NDI. Unlike NDI, which requires accurate model information in a closed-loop dynamic system, the INDI uses a state feedback structure. The general nonlinear control-affine system equation of an aircraft is as follows:

$$\begin{aligned}
\dot{x} &= f(x) + g(x)u \\
\mathrm{y} &= \mathrm{h(x)}
\end{aligned} \tag{16}$$

where $x \in \mathbb{R}^n$ represents the state vector, $u \in \mathbb{R}^m$ the input vector, $y \in \mathbb{R}^d$ the output vector, f and h are smooth vector fields, and $g \in \mathbb{R}^{n\times m}$is a control distribution function. Equation 16 is approximated using Taylor series expansion at the current state value x_o and the control input u_o

$$\begin{aligned}
\dot{x} \approx f(x_0) + g(x_0)u_0 + \frac{\partial}{\partial x}[f(x) + g(x)u]_{u_0,x_0}(x - x_0) \\
+ \frac{\partial}{\partial u}[f(x) + g(x)u]_{u_0,x_0}(u - u_0).
\end{aligned} \tag{17}$$

http://www.imavs.org/

x_o and u_o are incremental instances of x and u of the previous time step, respectively. The higher-order term is assumed to be negligible, so it is not expressed in Equation 17. Moreover, in general, the change in the one-step delayed state in aircraft application control is tiny and therefore negligible. Furthermore, the control input u is faster than the state x, and according to the time scale separation principle, it can be simplified as:

$$\dot{x} \approx \dot{x}_0 + g(x_0)\Delta u \tag{18}$$

where the increment in control input is defined as $\Delta u = u - u_o$. The incremental control law is derived from Equation 18 and assumes that the state variable equals the output value. Here, $\dot{x}_0$ is the angular acceleration in the previous time step, and $\dot{x}$ is expressed as ν in Equation 17. If the inverse of $g(x)$ exists, the control increment is represented as:

$$\Delta u = g(x_0)^{-1}(\nu - \dot{x}_0) \tag{19}$$

where ν is a dynamic characteristic that reflects the system design requirements and becomes the target dynamics. Based on the previous control input through Equation 19, the new control input is as follows:

$$u = u_0 + g(x_0)^{-1}(\nu - \dot{x}_0). \tag{20}$$

Substituting Equation 20 into Equation 17 gives:

$$\dot{x} = \nu. \tag{21}$$

The above equation shows that the existing system dynamic characteristics are removed, and characteristics according to the designer or system requirements are followed. In addition, the obtained control law does not depend on $f(x)$ and requires more nominal information about the system model than NDI. The sensor-based INDI is a method of directly acquiring $\dot{x}_0$ based on sensor measurement. Since the information measured from the sensor contains uncertain information that cannot be accurately expressed mathematically, it is a very robust control concept for compensating for model uncertainty.

3.2 Model-Based INDI

The model-based INDI is formulated by the same derivation procedure as sensor-based INDI and is a method of calculating the angular acceleration or secondary time-varying of the current system by mathematical modeling. The modeled dynamic $f_{mdi}(x_0) + g_{mdi}(x_0)u_0$ replaces $f(x_0) + g(x_0)u_0$ in Equation 17, and the derived control law is as follows:

$$\begin{aligned} u &= u_0 + g_{mdi}(x_0)^{-1}(\nu - \dot{X}) \\ &= u_0 + g_{mdi}(x_0)^{-1}(\nu - f_{mdi}(x_0) - g_{mdi}(x_0)u_0) \end{aligned} \tag{22}$$

where $\dot{X} = f_{mdi}(x_0) + g_{mdi}(x_0)u_0$ means the mathematical modeling equation of the system. In model-based INDI, the increment u_0 is removed, so the expression is simplified as:

$$u = g_{mdi}(x_0)^{-1}(\nu - f_{mdi}(x_0)). \tag{23}$$

The model-based INDI is called On-Board Model (OBM). The linear control law consists of linearized model information for each trim point in the system and a datasheet of trim state variables. A nonlinear system requires a relatively higher calculation than a linear system by mathematically expressing dynamic information.

3.3 Hybrid incremental nonlinear dynamic inversion

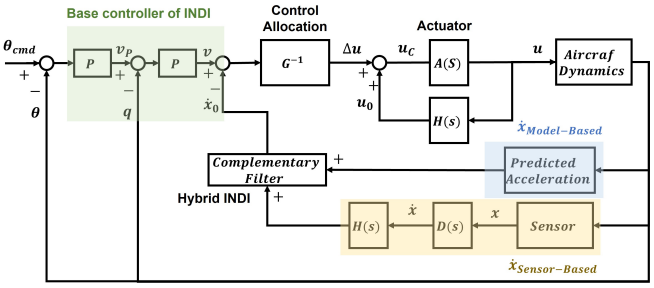


Figure 3: Control structure of the hybrid INDI control

A typical complementary filter reduces the effect of noise by combining multiple signals to use a wide frequency band of the signal. It is possible to generate a signal in which the noise characteristics of the low-frequency signal are subjected to high-pass filtered, and the high-frequency noise characteristics are subjected to low-pass filtered. In this paper, the sensor differentiation method and model prediction estimation method are considered to estimate acceleration or angular acceleration for dynamic system control. Both methods have their advantages and disadvantages. The model prediction estimation method is less noisy, has a little phase lag, and can compensate for the drawbacks of the sensor differentiation method. The sensor differentiation method can estimate the actual angular velocity or angular acceleration signal and does not depend on system model information. It is also possible to reflect disturbances and system uncertainties

$$\begin{aligned} \hat{\dot{x}} =& [\dot{x}_2 G(s) - \hat{\dot{x}} x G(s)][H(s)] + \dot{x}_1 \\ =& [x_2 - \hat{\dot{x}} x \frac{1}{s}][K_P + \frac{K_I}{S}] + \dot{x}_1 \end{aligned} \tag{24}$$

The Equation 24 can be rearranged as follows:

$$\hat{\dot{x}} = L_f(s)x_2 + H_f(s)\dot{x}_1. \tag{25}$$

where $L_f(s)s$ and $H_f(s)s$ are

$$L_f(s) = \frac{K_P s + K_I}{s^2 + K_P s + K_I} s, \quad H_f(s) = \frac{s^2}{s^2 + K_P s + K_I}. \tag{26}$$

4 SIMULATION AND RESULTS

4.1 Simulation conditions

As shown in Table 1, the simulation is performed using the trim conditions of the fixed-wing flight state. The attitude

http://www.imavs.org/

Table 1: Simulation conditions

Simulation time (dt:0.01 s)	$0 \sim 60s$
Initial condition	$\phi = 0$ deg $\theta = \alpha = 0$ deg $\psi = 0$ deg $\dot{p} = 0$ deg $/s$ $\dot{q} = 0$ deg $/s$ $\dot{r} = 0$ deg $/s$ $V = 25\,m/s$
Control surfaces	$\Omega_1 = 0$ $\Omega_2 = 0$ $\Omega_3 = 2292$ $\Omega_4 = 2292$ $\delta_{ev}^R = -1.43$ deg $\delta_{ev}^L = -1.43$ deg
Attitude command (Butterworth filter)	$\theta_{cmd} = 10^\circ$ $(20 \sim 30s)$ $\theta_{cmd} = -10^\circ$ $(30 \sim 40s)$
Gyro sensor noise (Gaussian Distribution)	$\ddot{x}_n(x_n) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x_n-\mu}{\sigma}\right)^2}$ $where\, \sigma = 1,\, \mu = 0$

command is a doublet command, and the Butterworth filter method is applied to the command to prevent actuator saturation. To compare the performance of INDI proposed in this paper, the sensing noise of the gyroscope is added. Simulations are performed in three scenarios. Depending on each scenario, actuator modeling is included, or noise is added to the angular velocity measurement. The PID, NDI, and INDI controllers are performed together on the same baseline control gain to compare the attitude tracking performance.

Table 2: Scenarios in Simulation

Scenarious	Actuator model	Angular velocity noise
Case1	**X**	**O**
Case2	**O**	**X**
Case3	**O**	**O**

4.2 Case1. w/o Actuator model, w/o angluar velocity noise

Figure 4 shows the results of case 1 is a scenario without an actuator model and angular velocity noise. The PID controller shows a significant offset for the angle command. Conversely, NDI shows excellent tracking performance. However, the NDI controller tracks the attitude with a constant offset from the beginning of the simulation time. INDI and HINDI have attitude tracking performance similar to NDI. However, the HINDI has some oscillation in the beginning. This phenomenon is considered an error occurring in the initial simulation condition when the actuator value of the previous step is fed back.

Table 3 compares the root-mean-square error (RMSE). The angle error has the smallest in the NDI, but the angular velocity error has the smallest in the HINDI. In case 1, since there is no actuator model and angular velocity noise, the NDI tracking performance that actively uses model information is excellent.

4.3 Case2. w/ Actuator model, w/o angluar velocity noise

Figure 5 represents case 2 with the actuator model and no angular velocity noise. Having an actuator model means it takes time for the actuator to move to the target angle. Therefore, the model uncertainty problem appears among the disadvantages of the NDI. In the case of NDI, large oscillations appear in the actuator from the beginning of the simulation, and if it is an actual flying vehicle, the actuator would be damaged. The attitude and angular velocity also exhibited significant vibrations, resulting in a greater decrease in altitude. In the case of the INDI, an undershoot occurs due to attitude tracking performance. However, it can be seen that the HINDI properly follows the doublet command, and the altitude drop is negligible without actuator saturation.

In Table 4, the attitude error of NDI shows a significant vibration in attitude and angular velocity. However, the angular velocity error is the largest compared to other controllers. In the case of the NDI controller, since significant vibrations continuously occur throughout the simulation, it is not a typical simulation result, so the system performance cannot be compared only with the RMSE. In contrast, the HINDI has an angular velocity error of less than 1 deg $/s$ and shows the highest performance.

4.4 Case3. w/ Actuator model, w/ angluar velocity noise

Figure 6 displays the simulation result considering the actuator model and angular velocity noise. The angular velocity noise causes the overall vibration. Since the NDI controller is greatly affected by model uncertainty and uses the model information, the angular velocity mixed with noise appears as a control output. In the case of the PID, the NDI, and the INDI, the attitude tracking performance is worse than that of case 2, and the actuator output also generates vibration. Conversely, the HINDI has improved attitude tracking performance, and the actuator output also has less vibration than other controllers. However, since the sensor-based angular velocity mixed with noise is used, the effect of vibration occurring throughout the simulation cannot be eliminated. Using complementary filters is that angular acceleration can be observed in both model-based and sensor-based methods. This structure showed a more robust performance against the uncertainty of the INDI model.

As shown in Table 5, HINDI shows improved attitude and angular velocity tracking compared to other controllers. The HINDI maintains robust performance while noise concerning angular velocity is appropriately canceled. It is considered

http://www.imavs.org/

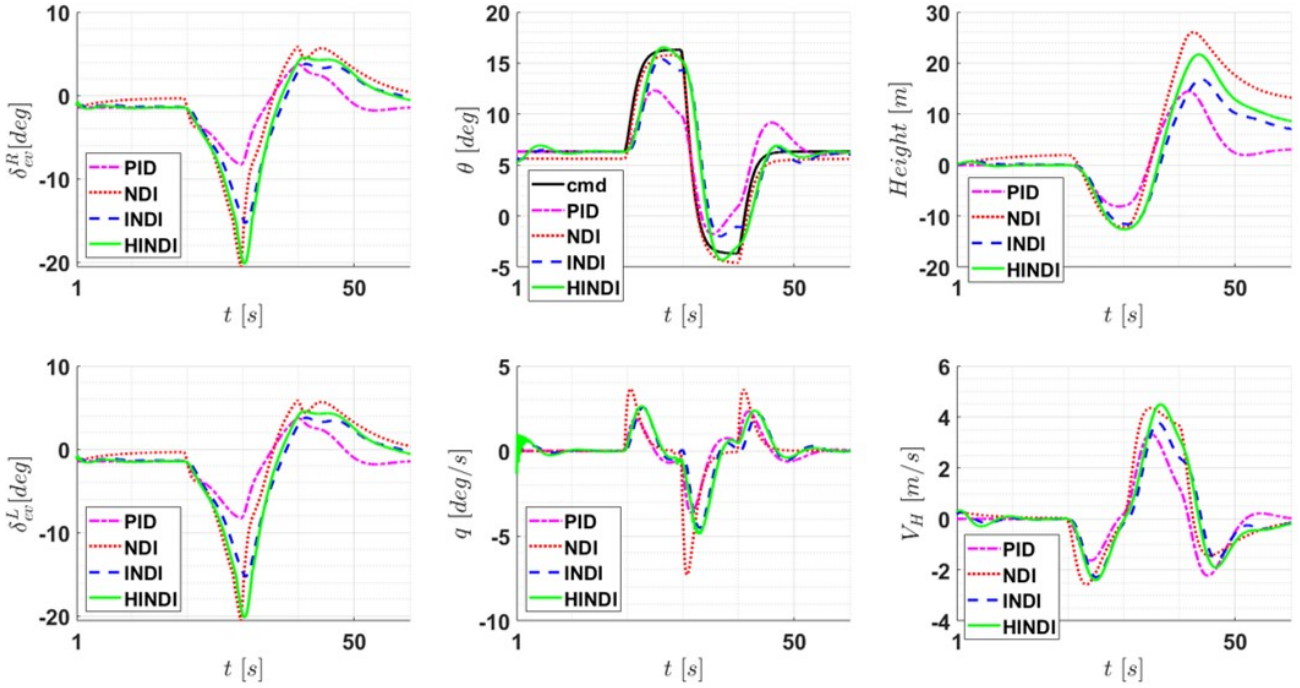


Figure 4: Simulation result-case1. w/o actuator model, w/o angular velocity noise

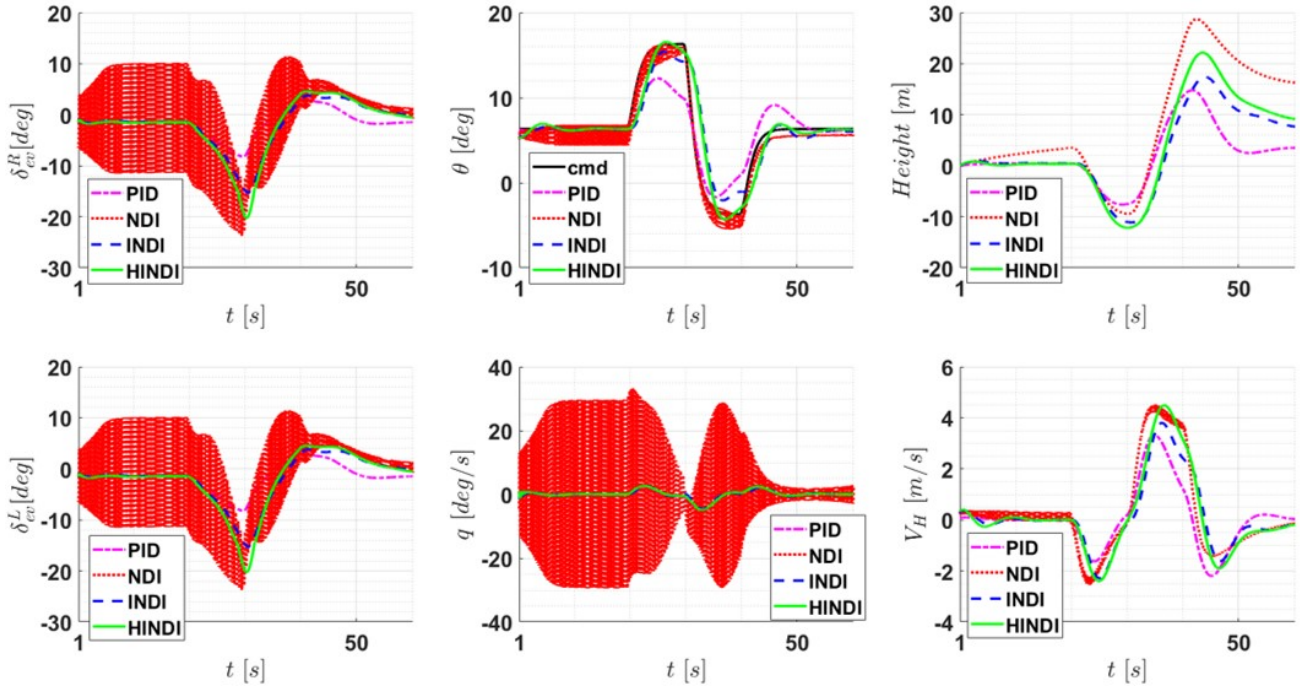


Figure 5: Simulation result-case2. w/ actuator model, w/o angular velocity noise

that it is unreasonable to use only RMSE to analyze the control performance of a system with a high vibration like case 2 and case 3.

http://www.imavs.org/

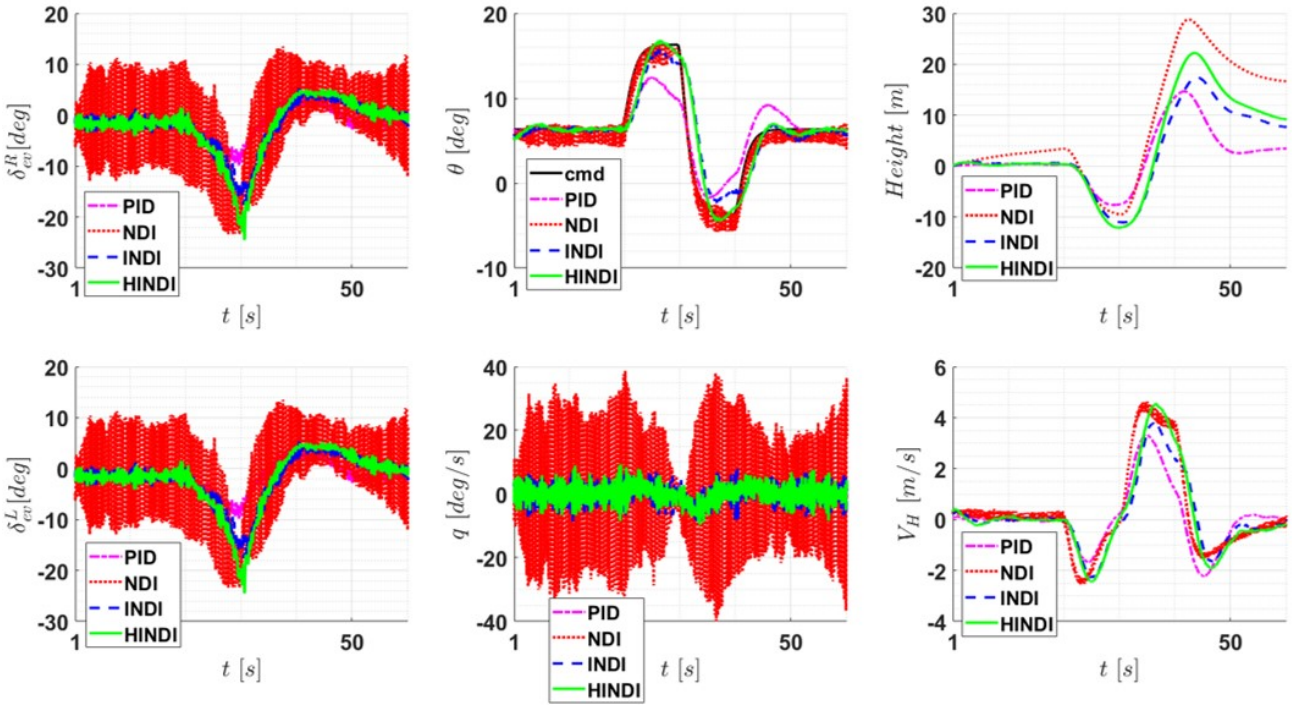


Figure 6: Simulation result-case3. w/ actuator model, w/ angular velocity noise

Table 3: RMSE result of Case1

RMSE	PID	NDI	INDI	HINDI	Unit
θ Tracking error	2.305	0.753	2.212	1.883	deg
q Tracking error	2.628	8.258	1.333	0.960	deg/s

Table 4: RMSE result of Case2

RMSE	PID	NDI	INDI	HINDI	Unit
θ Tracking error	2.312	1.015	2.243	1.884	deg
q Tracking error	2.616	16.532	1.387	0.963	deg/s

Table 5: RMSE result of Case3

RMSE	PID	NDI	INDI	HINDI	Unit
θ Tracking error	2.327	1.088	2.253	1.920	deg
q Tracking error	3.144	18.921	2.952	2.667	deg/s

5 Conclusions and Future work

To improve the noise or time-delayed angular acceleration of the existing INDI controller, in this paper, a HINDI controller was designed using a complementary filter. For numerical simulation, aerodynamic data and propulsion system modeling of a tiltrotor UAV were performed. Moreover, sensor-based and model-based INDI was discussed for acquiring angular acceleration signals of the existing INDI. Combining the advantages of the sensor-based and the model-based INDI, a complementary filter was applied to design the HINDI controller structure. The attitude tracking simulations of PID, NDI, INDI, and HINDI controllers were performed, and it was confirmed that HINDI had the best attitude tracking performance in the three simulation cases.

The proposed algorithm will be applied to a tiltrotor UAV to verify the robust attitude control in transition and fixed-wing flight modes in a future study.

Acknowledgements

This work was supported by the Korea Evaluation Institute of Industrial Technology (KEIT) grant funded by the Korea government (MTIE) (No.20016463).

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government(MSIT) (No. NRF-2021R1A2C2013363).

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government(MSIT)(No. 2021R1A5A1031868).

References

[1] Z. Liu, Y. He, L. Yang, and J. Han. Control techniques of tilt rotor unmanned aerial vehicle systems: A review.

http://www.imavs.org/

Chinese Journal of Aeronautics, 30(1):135–148, 2017.

[2] N. T. Hegde, V. I. George, C. G. Nayak, and K. Kumar. Design, dynamic modelling and control of tiltrotor uavs: a review. *International Journal of Intelligent Unmanned Systems*, 8(3), 2019.

[3] P Smith. A simplified approach to nonlinear dynamic inversion based flight control. In *23rd Atmospheric Flight Mechanics Conference*, 1998.

[4] 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, 2010.

[5] P. Acquatella, W. Falkena, E. J. van Kampen, and Q. P. Chu. Robust nonlinear spacecraft attitude control using incremental nonlinear dynamic inversion. In *AIAA Guidance, Navigation, and Control Conference*, page 4623, 2012.

[6] Barton Bacon and Aaron Ostroff. Reconfigurable flight control using nonlinear dynamic inversion with a special accelerometer implementation. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2000.

[7] Hong-Xing Deng, Xian-Bin Wang, and Ka Liu. Design of an angular acceleration boundary observer. *Journal of Harbin Institute of Technology*, 42:1504–1508, 2010.

[8] Wang Wanjun Zhang Weigong Li Xu. Inertia electrical emulation and angular acceleration estimation for transmission test rig. *Journal of Southeast University (Natural Science Edition)*, 42:62–66, 2012.

[9] Seppo J. Ovaska and Sami Valiviita. Angular acceleration measurement: A review. In *IEEE Instrumentation and Measurement Technology Conference*, 1998.

[10] Jari Pasanen, Olli Vainio, and Seppo J. Ovaska. Predictive synchronization and restoration of corrupted velocity samples. *Measurement*, 13(4):315–324, 1994.

[11] P. R. Belanger, P. Dobrovolny, A. Helmy, and X. Zhang. Estimation of angular velocity and acceleration from shaft-encoder measurements. *The International Journal of Robotics Research*, 17(11):1225–1233, 1998.

[12] W Blake. Prediction of fighter aircraft dynamic derivatives using digital datcom. In *3rd Applied Aerodynamics Conference*, 1985.

[13] B. M. Min, S. S. Shin, H. C. Shim, and M. J. Tahk. Modeling and autopilot design of blended wing-body uav. *International Journal of Aeronautical and Space Sciences*, 9(1):121–128, 2008.

[14] R. Czyba, M. Lemanowicz, Z. Gorol, and T. Kudala. Construction prototyping, flight dynamics modeling, and aerodynamic analysis of hybrid vtol unmanned aircraft. *Journal of Advanced Transportation*, 2018.

[15] P. J. Shao, W. H. Dong, X. X. Sun, T. J. Ding, and Q. Zou. Dynamic surface control to correct for gyroscopic effect of propellers on quadrotor. In *International Journal of Aeronautical and Space Sciences*, 2015.

http://www.imavs.org/

Recyclable bio-based composite Flax/Elium for UAV applications

M. Lefebvre[1,2]*, T. Pavot[1], R. Kiefer[1], M. Vedrines[1], N. Bahlouli[2]
[1]INSA Strasbourg, ICube laboratory, 24 Boulevard de la Victoire, Strasbourg, France
[2]Strasbourg University, ICube laboratory, 2-4 Rue Boussingault, Strasbourg, France

http://www.imavs.org/

ABSTRACT

Bio-based flax fiber reinforced composites (FFRC) are more and more used for land or aerial applications because of their good specific mechanical properties [1, 2, 3]. The use of those composites allows to design strong and light structures with minimum environmental impact. In this study FEA (Finite Element Analysis) simulations are executed on an autonomous long range drone (ELCOD drone; www.elcod.eu). A long range fuel cell powered drone was developed and built to do aerial pollution monitoring. The objective is now to replace the current glass fiber reinforced composite (GFRC) structure by a recyclable flax fiber structure. After exhaustive mechanical experiments at different scales of the composite (microscopic and mesoscopic), mechanical properties of the material obtained are computed into FEA simulations and load factor from 1 to 10 G is applied on the structure with a maximum permissible load of 7.5 G. The failure criterion studied is the Hashin factor. Simulations show that structure made with a fully bio-based composite has lower resistance compare to glass fiber structure and failure can occurs prematurely in compression. However, the use of FFRC offers a substantial gain weight. Hence, the ideal composition is an hybrid glass/flax composite or a carbon/flax fiber composite. The last part of this study presents simulation with a hybrid structure made with carbon fiber reinforced composite (CFRC) mixed with FFRC.

1 INTRODUCTION

Carbon fiber reinforced composites and glass fiber reinforced composites are suitable for UAV structure because they are lightweight and have excellent mechanical properties. The fixed wing drone Stork prototype has been developed and designed during the ELCOD INTERREG project with a conventional CRFC and GFRC structure. This drone features a brushless motor powered by a hybrid fuel cell and LiPo battery source. However, conventional composites if they are made with thermoset matrix (epoxy or polyester), are non recyclable and have a very high environment impact. For instance, the carbon footprint of CFRC is 12 times higher than GFRC (calculated in kg CO_2 eq/t) and the carbon footprint of GFRC is 6 times higher than FFRC [4]. Moreover, FFRC are good alternatives to petro-based composites because flax fibers have comparable mechanical properties with glass fibers for the strongest variety and have lower density. The objective is to investigate if recyclable bio-based FFRC are suitable for UAV applications and if it's possible to avoid the use of carbon fiber in the next generation of the Stork drone MkII. The first part of the study presents experimentation on the flax elementary flax fiber and the FFRC in quasi-static at different scale of the material. The objective is to find homogenization law to find the mechanical properties of FFRC from the mechanical properties of elementary flax fibers. The second part of this study presents simulations to design the drone structure and a comparison between a fully glass fibers structure and a fully flax fibers structure. For simulations the FFRC mechanical properties used are the ones determined with mechanical experiments. Moreover, a simulation with a CFRC wing spar and cap is simulated for comparison (see figure 7 for technical words).



Figure 1: ELCOD Drone Stork Prototype - 5 m wingspan; 16 kg without payload; MTOW: 25 kg

2 MATERIAL TESTING

2.1 *Quasi-static testing - Flax fiber*

Experiments in quasi-static are executed at the scale of the flax fiber and at the scale of the composite. The objective is to find homogenization laws to switch from the micro-

*Email address(es): martin.lefebvre@insa-strasbourg.fr

http://www.imavs.org/

scopic scale to the mesoscopic scale of the composite laminate. In the table 1, mechanical properties of flax fibers are compared with glass and carbon fibers. It can be noticed that the range of mechanical properties for flax fibers is wide as a consequence of the heterogeneous composition and structure of natural materials [3].

Fiber	Density [g/cm^3]	YM [GPa]	TS [MPa]	Price [$/kg]
Flax	1.4-1.5	30-80	350-2000	0.5-1.5
E-Glass	2.5	70-85	2000-3500	1.5-3
Carbon	1.4	230-240	4500-4800	10-20

Table 1: Mechanical properties of fibers (YM for Young's modulus and TS for Tensile Strength) [1, 5, 6].

In order to model the mechanical behavior of the composite, tensile test are executed on elementary flax fiber. In this study, flax tows are supplied by Eco-Technilin (Normandy, France). Elementary flax fibers are typically considered as a multiscale bio-composite because of their multi-component composition and their multi-scale structure. Elementary fibers have a diameter between 10 and 40 µm and are linked together in the stem of the plant with pectin to form a bundle [7, 8]. Technical fiber is the adjective uttered for a bundle of tens of elementary fibers (between 10 and 40 units). Each elementary fiber is composed of a thin external layer of 0.2 µm, called the primary cell wall, a thick secondary cell wall and a hollow in the center called lumen [9, 10, 7]. Cell walls are naturally bio-composed of cellulose microfibrils laid in spiral with various structures around the fiber axis and embedded in hemicellulose bonds and a non-cellulosic polymer (pectin and lignin) [11]. Those components are called bio-molecules. The spiral structure typically has an angle between 6 and 11° with the longitudinal axis [6, 7]. This angle is called microfibrils angle or MFA (see figure 2). Tensile tests on elementary fiber are based on the standards NF T 25-501-2. Tensile test are performed with an Instron 5944 with a load cell Instron 100 N. Elementary flax fibers are manually extracted from the bundle according to the standards defined and dried for 24 h at 40 °C. The fiber is glued in a frame before mounting: The elementary fiber is glued with cyanoacrylate on two parts of sanding paper (figure 3). This paper reduces the sliding risk of the sample during the tensile test [11]. For quasi-static tensile tests, 105 fibers are prepared and 71 are exploitable and results are plotted in the figure 4 and showed in the table 2.

Tensile stress strain curves of elementary flax fibers are plotted in the figure 4 and shows all the fibers tested and highlight the significant variability of the tensile behavior of elementary flax fiber. This hyper-elastic non-linearity of the tensile behavior of flax fiber can be explained by a reorientation of the cellulose micro-fibrils (micro-fibrils angle or MFA) during the tensile tests [11, 8].

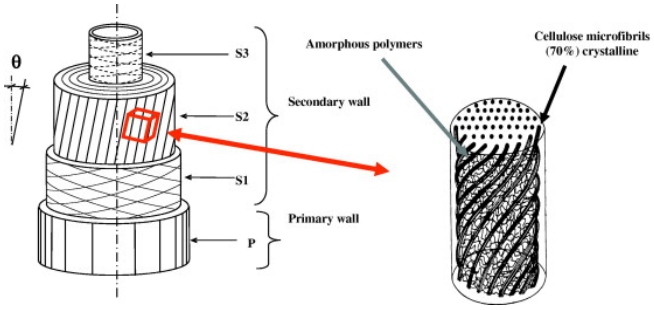


Figure 2: Schematic representation of one elementary flax fiber [11].

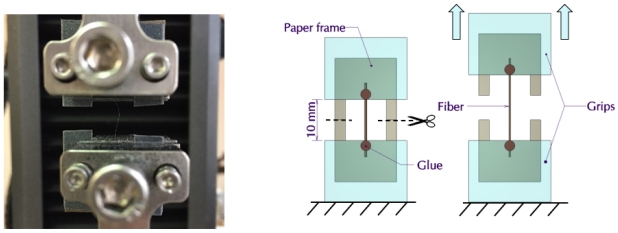


Figure 3: Experimental setup for tensile test on flax fiber.

In conclusion, flax fibers show very good mechanical properties, comparable to glass fibers for the Young's modulus, but with high variability. However neither glass or flax fibers can replace carbon fibers. But, considering the low density of flax fiber, that make them suitable for aerial applications.

2.2 *Quasi-static testing - Flax fiber reinforced composite*

Tensile tests are performed with an Instron 5969 with a load cell Instron 50 kN. Flax fibers are supplied by Eco-Technilin (Normandy, France) in the form of unidirectional plies (FlaxTape 110 g/m^2). The composite plates are then made by the vacuum infusion process at 0.9 bar with Elium resin at Arkema's technical center (Lacq, France). Elium matrix is a thermoplastic polymer with mechanical properties similar to epoxy. Moreover, this thermoplastic matrix can be processed by hand lay-up for composite parts. Laminated plates are made to test the different orientations of the

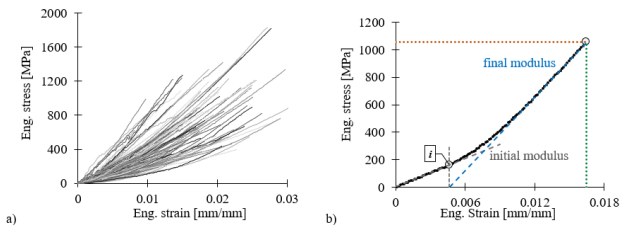


Figure 4: Experimental results - Tensile tests on : a) All fibers; b) One particular fiber - Increasing of stiffness.

Young's modulus [GPa]	Tensile strength [MPa]	Strain at failure [%]
61.5 (24.0)	805.2 (370.9)	1.72 (0.5)

Table 2: Averages mechanical properties for all fibers tested (Standard deviation).

composite in traction and compression: $[0]_{12}$ and $[0,90]_{3s}$ for traction (figure 5) and $[0]_{40}$ and $[0,90]_{10s}$ for compression. Cylindrical specimens are cut so that the fibers are coplanar to the cylinder axis for compression testing (figure 6). The chosen L/D ratio is 0.5 with a diameter of 8 mm and a thickness of 4 mm. This aspect ratio is commonly used in the literature to test under compression bio-based composites [12, 13]. This ratio is chosen according to ASTM E9-89 for compression testing. The fiber volume fraction of the composite is calculated as $V_f = 38 + / - 3\%$. The fiber volume is calculated by making smooth sections and measuring the area occupied by the fibers.

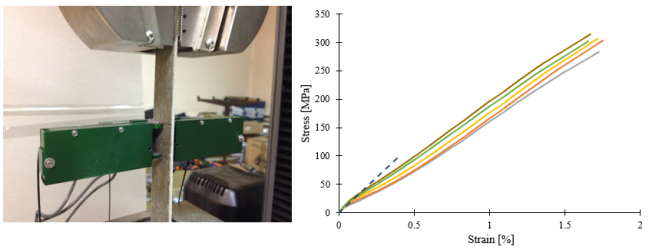


Figure 5: Tensile experiment and results on composite laminate [0]12

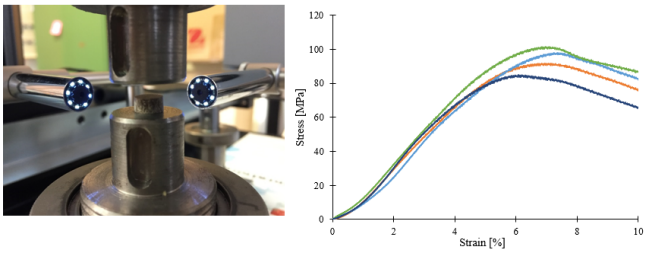


Figure 6: compression experiment an results on composite laminate [0]40

All the tensile tests and compression tests allow to get the mechanical properties of one Flax/Elium ply. Hence, the mechanical properties of the composite obtain with experimental results are sorted in the table 3:

$X_{T/C}$ is for the tensile strength and compressive strength respectively in the longitudinal direction. If the stress reaches this level in the composite, failure occurs. It can be noticed that the tensile properties of FFRC in tension are excellent and comparable to GFRC. However, mechanical properties

	Tension	Compression
E_l	24.8 GPa	2.1 GPa
E_t	3.7 GPa	1.2 GPa
G_{lt}	1.6 GPa	0.6 GPa
ν_{lt}	0.29	0.18
$X_{T/C}$	296.5 MPa	93.3 MPa

Table 3: Mechanical properties of one ply Flax/Elium

are lower in compression. In compression, the properties of the composite come mainly from the matrix. In tension, the properties come from the fiber [13].

2.3 *Homogenization from microscopic (elementary flax fiber) to mesoscopic (FFRC scale)*

In this part, homogenization laws are used to predict the mechanical behavior of FFRC from the mechanical properties of elementary flax fiber. Tsai with contiguity equations are used for homogenization and they are described by [14] :

$$E_l = E_{fL}.V_f + E_m.V_m \tag{1}$$

$$E_t = A\left[(1-c)\,B + c.C\right] \tag{2}$$

Where the constants A, B and C are :

$$A = 2\left[1 - \nu_f + V_m\left(\nu_f - \nu_m\right)\right], \tag{3}$$

$$B = \frac{K_f\left(2.K_m + G_m\right) - G_m\left(K_f - K_m\right)V_m}{\left(2.K_m + G_m\right) + 2\left(K_f - K_m\right)V_m} \tag{4}$$

$$C = \frac{K_f\left(2.K_m + G_f\right) - G_f\left(K_m - K_f\right)V_m}{\left(2.K_m + G_f\right) - 2\left(K_m - K_f\right)V_m} \tag{5}$$

$$K_f = \frac{E_{fL}}{2\left(1-\nu_f\right)}, K_m = \frac{E_m}{2\left(1-\nu_m\right)} \tag{6}$$

Composite are made of fibers and polymer matrix with different mixing ratios. V_f is the fiber volume fraction in percentage in the composite and V_m is the matrix volume fraction. G_f and G_m are the shear modulus for fiber and matrix respectably. c is the contiguity factor, with $0 \leq c \leq 1$. In the case of all of the fibers are in contact in the composite and continuity thus, $c = 1$. In the case of all the fibers are separated (or isolated) in the composite, thus $c = 0$. In the case of the study, this coefficient is difficult to determine. However, a value of $c = 0.1$ allow to minimize the error between experimental value and modeling value. In the table 4, it can be noticed that Tsai with contiguity model gives very low error with experimental values. Tsai with contiguity factor model allows to determine the mechanical properties of the composite by knowing the elementary properties of the matrix and

http://www.imavs.org/

the fibers. Thereafter we will use the calculated mechanical properties of these composites to perform our FEA simulations.

	Experimental	Tsai model	Error
E_l	24.8 GPa	25.4 GPa	2%
E_t	3.7 GPa	3.8 GPa	1 %
G_{lt}	1.6 GPa	1.6 GPa	0 %
ν_{lt}	0.29	0.28	2 %

Table 4: Mechanical properties of one ply Flax/Elium

3 DRONE SIMULATIONS

3.1 Geometry and boundaries conditions

Simulations are executed with Abaqus CAE. Half a drone is design to compute simulations. The lift calculated with Star CCM+ is then mapped on the wing surface of the drone, representing 250 N for a cruise speed of 25 m/s. Then the gravity is applied to all of the structure (figure 8 and 9). The lift coefficient is then changed to have a load factor from 1 to 10 G. The maximum limit for the UAV is define to 5 G according to White et al. [15], with a factor of safety equals to 1.5 based on the FAA regulations (FAR 25.303) is chosen. The maximum load factor on the structure is equal to 7.5. Stack sequence is created in Abaqus with the different composites. GFRC values come from literature and FFRC come from previous experiments and model. It has to be noticed that in all simulations the wing joiner of the drone is fully made with CFRC and the elevator is made with FFRC. The fabrics used for simulations are: a 110 g/m^2 for flax fiber, a 100 g/m^2 for glass fiber and a 80 g/m^2 for carbon fiber with theoretical fiber volume fraction of 45 %. The wing structure is described in the figure 7. The upper surface and lower surface are made with glass (GFRC) or flax (FFRC) twill fabrics and AIREX C70 PVC foam with the following layup : [(0,90),AIREX,(0,90),(+45,-45)]. The spar caps are made with glass (GFRC) or flax (FFRC) twill and unidirectional fabrics with the following layup : [0,0,0,(0,90),(+45,-45)]. The layup for the hybrid carbon / flax reinforced composite structure (CFRC / FFRC) is : [0,(0,90),(+45,-45)]. The high strength carbon fibers allow to reduce the number of unidirectional plies at 0° in the longitudinal direction for the spar caps.

3.2 Flax fiber structure vs conventional glass fiber structure

One of the main advantages for FFRC is a low density in comparison to GFRC because flax fibers have a lower density than glass fiber (see table 1). In the following study, FFRC is the name given for the structure 100 % made of FFRC, GFRC is the name given for the structure 100 % made of GFRC. For weight comparison, the conventional GFRC structure is defined as the reference. Simulations are then used to verify the structure integrity under various load factor and to evaluate

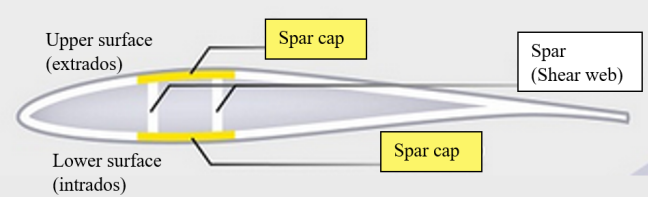


Figure 7: ELCOD drone wing section; structure with double spars.

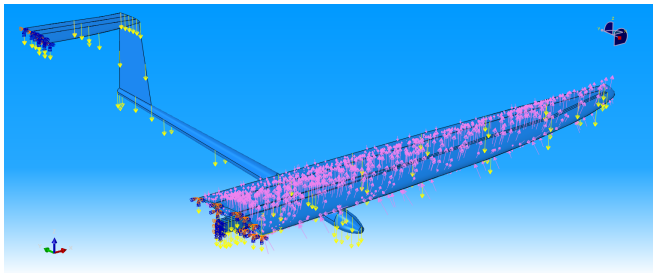


Figure 8: Boundaries conditions on the ELCOD MkII Stork drone - Gravity applied ; lift mapped ; symmetric link.

which structure is the most performing considering the gain weight. The most adequate value to design the drone structure is to use Hashin criterion. Indeed, this criterion distinguishes failure in tension and in compression and compares the structure stress to a maximum value [16]. Failure occurs when this ratio is over 1. If this factor equals to 1 in compression or tension, failure occurs in a composite ply (see equations 7 and 8). However, to design the drone structure, a safety coefficient s is used. The safety factor is usually set at 1.5. Hence, Hashin ratio are compared to 0.67 instead of 1. For every simulation, the maximum value selected is situated on the cap of the upper surface at the root of the wing for the compression and on the cap at the root of the wing of the lower surface for the tension. Because during cruise flight or important load factor (wing gust, stall recovery), bending occurs to the wing structure (see figure 9). Hence, the upper surface is solicited in compression and the lower surface is solicited in tension.

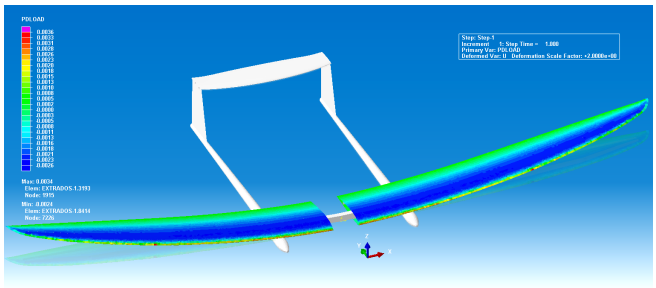


Figure 9: Lift pressure mapping - Overview of the wing bending.

http://www.imavs.org/

The spar of the wing is solicited in shearing.

$$\left(\frac{\sigma_{11}}{X_T}\right)^2 + \left(\frac{\tau_{12}}{S_{12}}\right)^2 \geq \frac{1}{s} = 0.67 \quad (7)$$

$$\left(\frac{\sigma_{11}}{X_C}\right)^2 \geq \frac{1}{s} = 0.67 \quad (8)$$

X_C is the compressive strength of the composite. X_T is the tensile strength of the composite. S_{12} is the shear strength of the composite. τ_{12} is the shear stress and σ_{11} is the stress in the longitudinal direction of the wing. Due to bending, the preponderant stress is in the longitudinal direction 11 of the wing (see figure 9). Results for FFRC and CFRC are plotted in the figure 10 and 11. In this study, for all composites laminates, fibers volume fraction is equals to 45 %. First of all, the use of a fully flax fiber structure allows to reduce weight to 11 % in comparison to a fully glass fiber structure for the total drone weight. For comparison, a value of 10 % of weight saving is also calculated in the study of Clifford et al. [2] for a wind blade made with FFRC. For both cases, it is clear that composite are stronger in tension, hence Hashin tension factor is lower than Hashin compression factor. Those figures show that until 10 G, the 100 % GFRC structure is under the limit in traction and compression. However, the 100 % FFRC structure is well under the limit in traction but over the limit in compression. The simulations show that failure will probably occur in compression with the 100 % FFRC structure around 5.5 G. The use of FFRC allows to reduce weight, but on the other hand, limits the load factor range on the structure.

3.3 Structure optimization: Hybrid composite

Regarding to the low mechanical properties of FFRC in compression two hybrid structures are tested. The objective is to use the benefit of low density of FFRC and avoid important stress in compression for FFRC. The first hybrid structure is made of FFRC for the spar of the wing and the lower surface (intrados) of the wing and made of GFRC for the upper surface (extrados) of the wing. As see previously, the upper surface is the part solicited in compression during cruise flight or for an important positive load factor. GFRC / FFRC is the name for the hybrid structure with lower wing surface made with FFRC and upper wing surface made with GFRC.

Results for hybrid composite GFRC / FFRC are plotted in the figure 10 and 11. It is clear that the hybrid structure allows both advantages of weight saving and good mechanical properties: The hybrid structure allows to stay under the limit in tension and compression and also allows a weight reduction to 10 %. At 5 G, the bending of the wing tip is at 161 mm for the FFRC structure, 126 mm for the GFRC structure and 155 mm for the GFRC / FFRC structure.

3.4 Hybrid composite with carbon fiber

Carbon fibers are well-know for light weight and high mechanical properties and are massively used for aerial applications [1]. The last simulation is performed with a hybrid

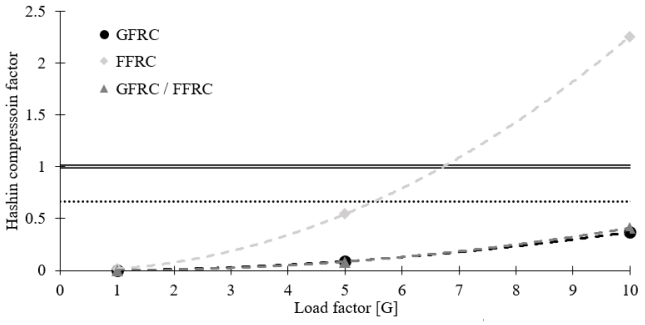


Figure 10: Hashin factor in compression function of the load factor; Limit value to 1 and 0.67 with safety coefficient.

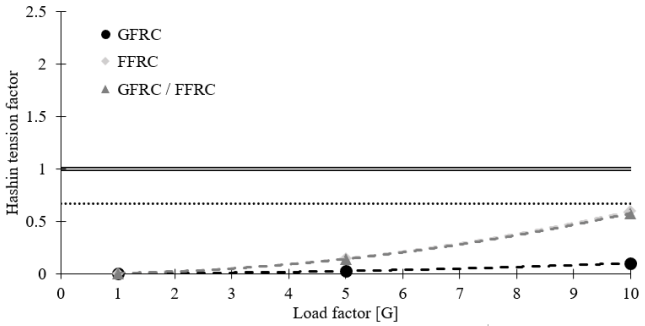


Figure 11: Hashin factor in tension function of the load factor; Limit value to 1 and 0.67 with safety coefficient.

CFRC / FFRC. This hybrid composite simulated for comparison is created with lower and upper surface skin made of FFRC and a wing spar and caps made of CFRC. This last simulation is called CFRC / FFRC. The figure 12 reveals that the use of CFRC for the spar and caps allows a maximum load factor of 17.5 G on the structure. Moreover, it allows a gain weight of 15 % compare to a fully glass fiber structure. However, the footprint of CFRC is 17 times higher than GFRC. In this case, the use of a hybrid CFRC / FFRC structure can be questionable if the objective is to reduce the footprint of material for UAV. However, the hybrid composite CFRC / FFRC has also many advantages: Flax fibers and carbon fibers have a similar density and a similar elongation at failure. A hybrid composite allows the flax to increase vibration absorption in the very stiff and strong carbon structure. Moreover, the carbon fibers are only used for the small surfaces of the caps. Then, the footprint for the use of carbon fiber is reduced.

The table 5 shows that the structure made of GFRC is the heaviest and the hybrid CFRC / FFRC structure is the lightest. The hybrid structure allows an important weight reduction to 15 % in comparison with the 100 % GFRC structure. The drone weight corresponds to the weight of both wing, wing joiner, elevator and twin booms. The weight of the fuselage and equipment (batteries, motor) are not taking into account.

http://www.imavs.org/

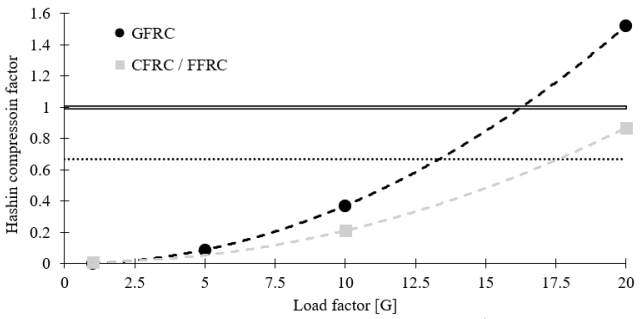


Figure 12: Hashin factor in tension function of the load factor - GFRC vs CFRC / FFRC.

Drone structure	Drone weight
GFRC	8.4 kg
FFRC	7.5 kg
GFRC / FFRC	7.6 kg
CFRC / FFRC	7.1 kg

Table 5: Weight comparison for several composites structures.

	1 G	5 G	7.5 G
Drone structure	Hashin factor - Compression		
GFRC	0.002	0.088	0.205
FFRC	0.016	0.545	1.257
GFRC / FFRC	0.003	0.085	0.215
CFRC / FFRC	0.0024	0.071	0.118
Drone structure	Hashin factor - Traction		
GFRC	0.002	0.088	0.053
FFRC	0.004	0.147	0.332
GFRC / FFRC	0.004	0.139	0.399
CFRC / FFRC	0.005	0.016	0.035

Table 6: Hashin criterion values for several composites structures.

4 CONCLUSION

The table 6 summarizes all the results for Hashin ratio for all structures simulated. The results show that even with the safety margin, the used of flax fiber as reinforcement don't causes premature failure on the structure in traction. And structure can be design in with flax or glass fibers. However, in compression, the Hashin factor is above 1 with a fully flax fiber structure at 7.5 G. The Hashin factor is equals to 1 around 6.8 G and equals to 0.67 around 5.4 G. If the FFRC structure is used, then the maximum load factor has to be reduced to 4.5 G. Previous flight tests with the Stork drone prototype show that this level of load factor can be easily reach during special aerial manoeuvres. The table 6 show that the used of carbon fibres for the spar and the caps and flax fibers for the skin allows the lowest Hashin ration in compression and traction for all load factors. Moreover, it allows a lighter composite structure.

It has been proven in this study that the use of FFRC offers a weight gain for UAV structure around 10%. However, FFRC has to be carefully used with compressive stress due to low mechanical properties in compression. For composite structure, the optimum is to design hybrid structures to benefits of the advantages of petro-based composites and bio-based composites. Another limit to the use of FFRC is the variability of the mechanical properties. The single fiber mechanical properties variability is reduced at the scale of the composite thanks to average effect with thousands fibers. However, is some industrial applications where low variability is required, the used of FFRC can be limited. Research has to be done to reduce the variability of natural fibers and allow their used for industrial applications. Concerning the study, the following step is to build a hybrid FFRC / GFRC wing and do experiment on the structure to validate the numerical model. The objective will be to fix strain gauges in the wing laminate and do bending tests on the wing and simulate a 5 and 7.5 G loading on the structure. The strain gauges will allow to compare the strain with the simulation and the strain in the real wing. With simulations, it has been proven that a fully FFRC structure is limited for long range UAV applications. However, a wing made of carbon fiber for the wing spar and flax fiber for the upper and lower surface will have excellent mechanical properties with a minimum weight. The drawback of this structure is the very high carbon footprint due to the use of carbon fibers. A good balance to design UAV, is to use hybrid composites GFRC / FFRC or CFRC / FFRC that will have enough strength for high load factor and will allow a gain weight between 10 and 15 %. However, when designing hybrid structure, the recyclability of the composite has to be investigated. If the thermoplastic matrix is theoretically recyclable, mixing FFRC with GFRC or CFRC could make the composite more complicated to recycle finally. One better solution is to design hybrid structure but without mixing material on same parts an allow easy separation of parts at the end of the cycle of life of the structure.

ACKNOWLEDGEMENTS

The authors thank Strasbourg University and INSA Strasbourg for funding this study.

REFERENCES

[1] D. Gay. Composite material design and applications. *CRC Press*, 2016.

[2] Mike J. Cliffor Darshil U. Shah, Peter J. Schubel. Can flax replace e-glass in structural composites? a small wind turbine blade case study. *Composites: Part B*, 52:172–181, 2013.

http://www.imavs.org/

[3] Opukuro David West R. A. Pethrick Mark C. Symington, W. M. Banks. Tensile testing of cellulose based natural fibers for structural composite applications. *Journal of composite materials*, 0(0), 2008.

[4] Martha Barth Niels de Beus, Michael Carus. Natural fibres show outstandingly low co2 footprint compared to glass and mineral fibres. *nova-Institut GmbH, press release*, 2019.

[5] CES EduPack. *Granta Design*, 2015.

[6] Bijoyee Sarker Khandakar M. Nahiun Ruhul A. Khan Fatin I. Mahir, Kamrun N. Keya. A brief review on natural fiber used as a replacement of synthetic fiber in polymer composites. *Materials Engineering Research*, 1(2):88–99, 2019.

[7] C. Goudenhooft. Multi-scale characterization of flax stems and fibers : structure and mechanical performances. *Materials*, 2018.

[8] Breard Joel Bourmaud Alain Davies Peter Baley Christophe, Gomina Moussa. Variability of mechanical properties of flax fibres for composite reinforcement. a review. *Industrial Crops and Products*, 145, 2020.

[9] Moussa Gomina Christophe Baley Laurent Bizet Joël Bréard Karine Charlet, Jean-Paul Jernot. Morphology and mechanical behavior of a natural composite: The flax fiber. *ICCM International Conferences on Composite Materials*, 2007.

[10] C. Morvan J.P. Jernot M. Gomina J. Breard K. Charlet a, C. Baley. Characteristics of hermès flax fibres as a function of their location in the stem and properties of the derived unidirectional composites. *Composites: Part A*, 38(8):1912–1921, 2007.

[11] C. Baley. Analysis of the flax fibres tensile behaviour and analysis of the tensile stiffness increase. *Composites: Part A*, 33(7):939–948, 2002.

[12] Chong Zhang Zhiqiang Zhang Dayong Hu, Linwei Dang. Mechanical behaviors of flax fiber-reinforced composites at different strain rates and rate-dependent constitutive model. *Materials*, 16(6):1–17, 2019.

[13] Fangxin Wang Wenyan Liang Kai Zhang, Yongyang Sun and Zhenqing Wang. Progressive failure and energy absorption of chopped bamboo fiber reinforced polybenzoxazine composite under impact loadings. *Polymers*, 12(8), 2020.

[14] Groun Brahim Brek Samir, Meddour Belkacem. Numerical modeling of the effects of fiber packing and reinforcement volume ratio on the transverse elasticity modulus of a unidirectional composite material glass / epoxy. *Revue des Composites et des Materiaux Avances*, 30(5):203–210, 2020.

[15] Villalba Jorge Martinez Diego Pardo Gerardo Cerón-Muñoz Hernán White Philippe, Vaca John. Redesign of an experimental unmanned aerial vehicle (uav) for its conversion into an operational laboratory platform. *International Journal of Applied Engineering Research*, 13(21):16438–16446, 2018.

[16] Xiang Zhang Jikui Zhang. An efficient approach for predicting low-velocity impact force and damage in composite laminates. *Composite Structures*, 130:85–94, 2019.

http://www.imavs.org/

Development of A Mission-Tailored Tail-Sitter MAV

Luiz F. T. Fernandez*[1,2], Murat Bronz †[1], Nathalie Bartoli‡[2], and Thierry Lefebvre§[2]

[1]ENAC, Université de Toulouse, F-31400 Toulouse, France
[2]ONERA/DTIS, Université de Toulouse, Toulouse, France

Abstract

Vertical takeoff and landing (VTOL) vehicles are among the most versatile UAVs, appropriate for various missions. Given that there are still open challenges regarding the VTOL design, this paper presents the full development and test cycle of a tail-sitter. IMAV 2022 competition rules are used to define the mission objective and constraints. A multidisciplinary design and optimization strategy is defined with the goal of maximizing competition score considering design, manufacturing, and competition constraints. While still being a work in progress, the resulting vehicle is designed to fly at 18 m/s, for around 14 minutes while carrying 200g of payload and weighting approximately 750g.

1 Introduction

Unmanned aerial vehicles (UAVs) have been object of study of several different research areas in the last decades. One can attribute this fact to two main reasons. From one side, such kind of vehicle can potentially be used for different missions, both civil and military. On the other hand, the technology poses a big variety of challenges to the aerial robotics community, which has primarily emerged from aerospace and robotics engineering fields. This community leverages from the fact that these vehicles can be designed and built with minimal cost, making it very attractive for real world testing and deployment. On the application side, there has been studies to use drones for fire fighting [1, 2], for searching marine debris [3], and even for covid mitigation [4]. Along with several studies regarding perception [5, 6], control is arguably one of the most studied fields within the UAV community [7, 8, 9] just to name a few. As for the design aspect of UAVs and specially micro aerial vehicles (MAVs), fewer studies have been published. Wang et al [10] presented a preliminary design methodology for small tail-sitters that considers weight and power models. Holsten et al. [11] presented the design strategy and wind tunnel test results for a multi-purpose tilt-wing platform. Vogeltanz [12] presented the design and analysis of the mini-UAV tail-sitter named "V-TS", including 2D and 3D high fidelity aerodynamic analysis and flight dynamics modeling. A very extensive review on design and flight control techniques is presented by Ducard and Allenspach [13]. This work tackles both vehicle design and control simultaneously, with the goal to obtain the most suitable vehicle to operate a given mission. In this context, MAV competitions are interesting as they pose well defined missions, suitable for testing and comparing design strategies.



Figure 1: Sequential imagery of vertical take-off and transition of the finalized prototype with a close-up picture.

The objective of this paper is to describe the full development cycle of a MAV, shown in Figure 1, providing the readers the possibility to understand our design decisions from the first iteration, how such decisions affected the initial tests, and how such tests affected our design process, in a spiral cycle. We present the initial design strategy, followed by manufacturing technique, and flight testing execution and results. Then another iteration is performed applying the knowledge obtained with initial flights and wind tunnel tests.

In Section 2, we explain the IMAV 2022 competition rules and score calculation. Our design philosophy and optimization strategy is detailed in Section 3. In Section 4, we show our preliminary analysis of the chosen configuration and candidate designs. Section 6 presents the manufacturing techniques and Section 7 details our test campaigns. Section 8 then uses the gathered knowledge to refine the MDO process, with the resulting vehicle, named *Falcon*, being presented in Section 9 along with flight test results. We show our conclusions and perspectives in Section 10.

2 Competition Rules and score analysis

The IMAV 2022 "Delivery far and fast" competition will be held in the Netherlands. The overall mission objective is to design and operate an MAV capable of carrying a specific amount of payload in a designated outdoor flight field. The

*luiz.tiberio@onera.fr
†murat.bronz@enac.fr
‡nathalie.bartoli@onera.fr
§thierry.lefebvre@onera.fr

http://www.imavs.org/

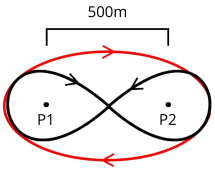


Figure 2: Schematics of competition track and trajectories

score is defined as:

$$score = \frac{N_{laps}(D+1)AP}{W_{vehicle}/W_{max}} \quad (1)$$

Table 1 describes each component of Eq. (1).

Parameter	Definition
N_{laps}	number of laps
D	1 if landed with payload and 0 otherwise
A	Autonomy level
P	Payload point factor
$W_{vehicle}$	Vehicle weight
W_{max}	Maximum allowed weight of 5kg

Table 1: Score factors

Given that P1 and P2 are obstacles located at $500m$ from each other, a lap can be defined in the following order: visit the line orthogonal to the line P1P2, pass through P1, visit the line orthogonal to the line P1P2 again, and pass through P2. Figure 2 shows a representation of the mission and two compliant trajectories (black and red lines). The autonomy factor (A) can assume the values: 1 (remote piloted UAV through video link), 2 (autonomous flight control), and 5 (autonomous mission control). The point factor (P) is defined as a function of the payload, as shown in Table 2.

Payload package mass [g]	P (Points factor)
100	1
200	2
500	3
1000	4

Table 2: Payload and Points factor

Throughout this paper, we will assume that the mission will be fully autonomous ($A = 5$), and no additional constraints for landing will be used to handle the parameter "D". The overall mission time is constrained to 30 minutes, including not only the full flight, but also the time expended to put the vehicle in the flight arena and turn on the electronics. So, a maximum flight time of 23 minutes is assumed.

From Eq. (1), we can observe that the mission score is mainly driven by the number of laps (N_{laps}) and vehicle weight ($W_{vehicle}$). As the number of laps is mainly a function of flight velocity, increasing this factor might imply in greater propulsion, battery and structural weight, which in turn is reversely proportional to the score. Thus, there is an implicit trade off in the score equation with respect to flight speed and vehicle weight. Such particularity makes the selection of design point more complicated when using traditional design techniques, as too much emphasis could be given to only one aspect of mission score. We used a multidisciplinary optimization (MDO) approach that allowed us not only to better explore the design space and the trade off, but specially to find the optimal design with respect to the score.



(a) FoxTech H-wing (b) Transition Robotics QuadShot

Figure 3: Baseline designs

3 DESIGN PHILOSOPHY

The initial hypothesis for this process was that a flying wing tail-sitter with four motors and no aerodynamic control surfaces was the most suitable configuration for this mission. Flying wing MAVs tend to have lower weights, and the use of four motors would allow for vectorized thrust based control, which in turn removes the necessity of control surfaces and additional servos. Figure 3 shows two MAVs used as baseline for our design process, the H-wing by Foxtech [1], and the QuadShot [2] by Transition Robotics.

The employed MDO approach was formulated to maximize the competition score by varying wing planform, angle of attack, and battery mass. Figure 4 shows the extended design structure matrix, as proposed by Lambe and Martins [14], created using WhatsOpt [15]. The optimization process consists of a multidisciplinary analysis (MDA) module and the mission score calculation. The MDA is a non-linear system of equations of three different disciplines: aerodynamics, mission analysis, and weights.

3.1 Aerodynamic analysis

OpenAerostruct [16] (OAS) was used for the aerodynamic analysis. It uses a vortex lattice method (VLM) to calculate aerodynamic coefficients. The MH45 airfoil was selected for the first iteration of the analysis. Table 3 shows that the payload dimensions, stipulated by the competition, are representative when considered to general MAV dimensions, so its drag can not be disregarded. Even if a greater payload tends to increase the score when considering the points factor (P),

[1] https://www.foxtechfpv.com/
[2] https://transition-robotics.com/pages/projects

http://www.imavs.org/

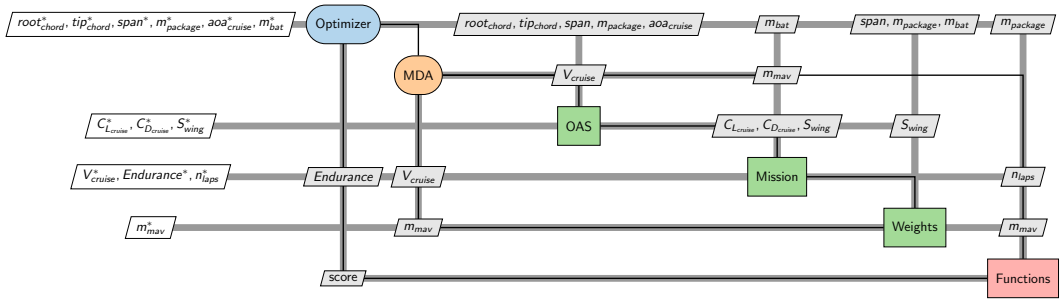


Figure 4: Extended design structure matrix of the tail-sitter process

Payload mass $[g]$	Dimensions lxbxh $[mm]$
100	100x60x40
200	100x80x60
500	150x100x70
1000	145x130x110

Table 3: Payload dimensions

this choice would directly lead to greater weights, both structural and battery, and drag. The payload drag coefficient ($C_{D_{Payload}}$), which is a rectangular body, was fixed to 1.2 according to Carvill [17]. Summarizing, the aerodynamic module calculates the total lift and drag coefficients for a given wing geometry, payload mass, and flight velocity.

3.2 Mission analysis and sizing

The mission module calculates flight speed, range, endurance, and number of laps for a given set of aerodynamic coefficients, wing area, MAV total mass, and battery mass. The cruise velocity (V_{cr}) is calculated to reinforce that lift is equal to weight in non-accelerated flight as:

$$V_{cr} = \sqrt{\frac{m_{mav} g}{0.5 \rho S C_L}} \tag{2}$$

where m_{mav} is the vehicle mass, g the gravity acceleration, ρ the air density, S the wing reference area, and C_L the lift coefficient in cruise condition. The flight range (R) can be obtained for a given battery mass (m_{bat}) and vehicle mass as:

$$R = \frac{m_{bat}}{m_{mav} g} B_{SE} \eta \frac{C_L}{C_D} \tag{3}$$

where C_D is the drag coefficient during cruise, the battery specific energy (B_{SE}) was assumed to be $140 \frac{Wh}{kg}$ and overall propulsive system efficiency η was assumed to be 0.32. Flight endurance (E) is then:

$$E = \frac{R}{V_{cr}}. \tag{4}$$

In order to obtain the number of laps, the lap distance was decomposed into two parts of straight 500m level flight, and two turns. The turn radius was calculated as:

$$T_R = \frac{V_{cr}^2}{g\sqrt{n^2 - 1}} \tag{5}$$

where the load factor (n) was fixed in 1.5, which corresponds to an approximately 50°bank angle. The number of laps was then obtained as the integer part of the ratio between flight range and lap distance:

$$N_{laps} = \lfloor \frac{R}{500 + 500 + 2\pi T_R} \rfloor \tag{6}$$

3.3 Weights module

The MAV total weight is obtained as a sum of fixed electronics, battery, spar, and structural weight. Table 4 shows the weight breakdown of fixed components.

Component	Mass $[g]$
Autopilot	12.0
Xbee module	5.2
Speed controller	20.7
RC receiver	7.7
Motor (each)	17.0
Motor arm (each)	35.0
GPS module	50.0
Cables	30.0

Table 4: Fixed weight breakdown

Using previous experience with 3D printed structures from[18], we chose to use this manufacturing technique with carbon tubular spars. We used high and low quality existing printed pieces to obtain an average of the surface weight as a function of its area. The obtained results are shown in Table 5. The wing spar weight was calculated considering 5mm carbon tube. Previous experience shows that instead of using one single spar in the full span, the usage of two spars with 50% span length each provides better rigidity, restricts the rotation motion, and ease component assembly without compromising the wingtip structural integrity or increasing weight.

http://www.imavs.org/

Structural rigidity	density of area $[g/m^2]$
High	2485
Low	941
Average	1713

Table 5: 3D printed structure weight as function of its area

3.4 MDO implementation

The MDO loop was implemented using OpenMDAO [19] and all the models were implemented in Python. The non-linear Gauss-Seidel method was selected for solving the MDA and we used IPOPT (Interior Point OPTimizer) from PyOptSparse [20] as the optimizer. Table 6 shows the design variables, objective, and constraints. For the optimization, the mission score is calculated as:

$$\text{score} = N_{laps}\frac{P}{m_{mav}g} \tag{7}$$

	Function/Variable	Lower	Upper
Maximize	score		
With respect to	Root chord	0.09 m	0.4 m
	Tip chord	0.05 m	0.4 m
	Span	0.3 m	1.0 m
	cruise α	$-10°$	$10°$
	Payload	0.1 kg	$1kg$
	Battery mass	0.01 kg	1.00 kg
Subject to	Endurance		23 min

Table 6: MDO problem formulation

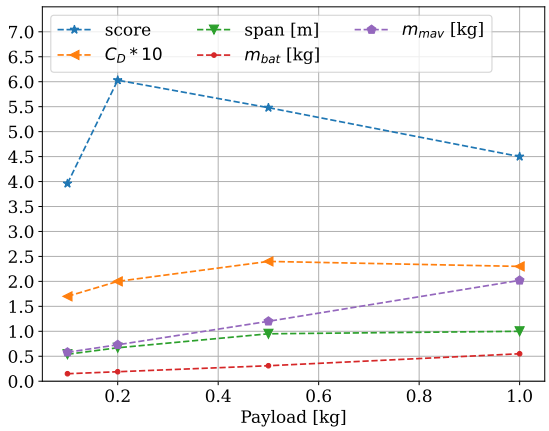


Figure 5: Maximized score, wing span and total drag for the allowed payloads.

Design variables	Value
Root chord	0.09 m
Tip chord	0.05 m
Wing span	0.66 m
Cruise α	$8.73°$
Battery mass	0.19 kg
Coupling variables	**Value**
Wing area	0.0467 m^2
Cruise speed	17.7 m/s
Cruise C_L	0.91
Cruise C_D	0.20
Flight time	23 min
Number of laps	21
MAV mass	0.74 kg

Table 7: Optimal MAV characteristics

4 Design Analysis

We started our analysis by evaluating how the competition score is effected by the different allowed payloads, shown in Table 3. For that, the MDO was executed four times, each of them for a fixed payload value. Figure 5 shows the characteristics of the obtained vehicles. It can be noted that choosing the smaller payload possible (100g) would lead to a non-competitive score, while the best values were found for the higher payloads. As it would be expected, the increase in payload mass also leads to bigger wing areas, which in turn increases the drag in cruise. The biggest payload designs are penalized by the drag generated by the bigger fuselage and wing. Figure 5 also shows that the observed difference in drag generates a significant difference in terms of battery and total weight.

This analysis showed that using $200g$ of payload mass for the first design iteration will be beneficial. Table 7 shows the characteristics of the first optimized MAV.

5 Flight Control System

We used Paparazzi[3] system [21], which allows for the automation of flight tasks up to complete autonomous flight for a designated route. The implemented INDI control law, successfully applied to tail-sitter in [9], was employed. The gains were estimated based on previous experience with tail-sitter and then refined after flight experiments. For the competition, the flight trajectory can be defined through Paparazzi's flight plan, and all the flight phases can to be fulfilled autonomously including safety measures, such as low-battery and GPS-lost mitigation maneuvers.

6 Manufacturing

One of the main objectives of this work is to showcase the iterative refining of the MDO process of the design for IMAV-2022's mission profile. Therefore, an agile manufacturing technique is chosen as proposed in Bronz et al. [18]

[3] https://wiki.paparazziuav.org/wiki/Main_Page

http://www.imavs.org/

using 3D printing. Such strategy allows for rapid manufacturing of different parts, which in turn enables faster design iterations.

7 Flight Tests

An iterative flight test procedure was followed, where the complexity of the mission increases according to the outcome of each test.

7.1 Hovering tests

The first hover test was conducted using only the main body of the vehicle inside ENAC's flight arena, as shown in Figure 6, while waiting for the manufacturing of the wing pairs. The gathered data were used to fine-tune the efficiency matrix of the INDI [9] controller for hover phase and proved that the vehicle is capable of stable hover with the designated payload capacity. The wings were then assembled for the second hover test.

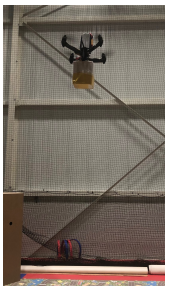


Figure 6: Hover test

7.2 Outdoor flight tests

After successfully performing the hovering test, the vehicle was tested in an outdoor flight field. Throughout the first day of tests, six successful non-autonomous flights were performed, with increasing level of difficulty starting from simple hover to the full flights with takeoff, cruise, and landing. Figure 7 shows the trajectory of the third flight, whose data will be shown from now on. After this approximately

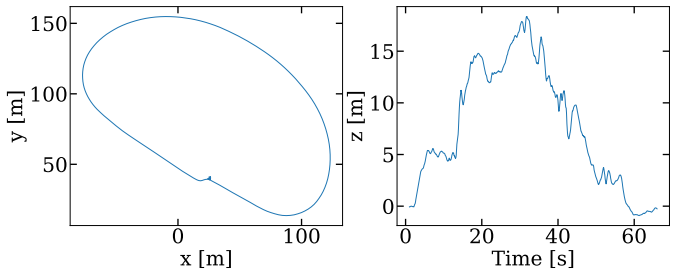


Figure 7: Experimental flight trajectory in XY plane and altitude.

one minute flight, the motors were overheated, indicating that an overload could be occurring. By analyzing Figure 8, that shows power consumption throughout the flight, it is indeed possible to note that even during cruise the motors are heavily loaded and no significant difference was observed between hover and cruise, which should not be the case for a winged vehicle. With such high values of power, the vehicle would

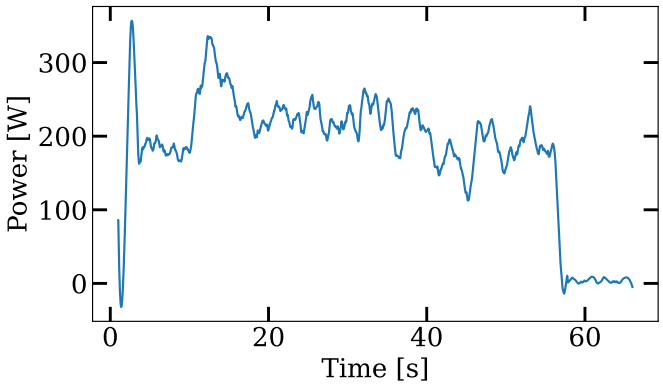


Figure 8: Experimental flight power consumption

not be able to perform the proposed 23 minutes mission. So it became necessary to investigate the source of such power consumption, with two main candidates:

- The set motors-propellers could be inadequate in terms of allowing for high range cruise and control authority. The only possibly suitable propellers available at that moment were the GS 5.0x4.5, but smaller diameters could increase cruise efficiency.
- The predicted drag polar could be overly optimistic for both lift and drag.

7.2.1 Propulsion system testing and autonomous flights

In order to compare different sizes of propellers and its impact on the performance, a new batch of outdoor tests was prepared and executed. We performed two fully autonomous flights, with the same trajectory and setpoints of 18 m/s and 30 m for cruise velocity and altitude, respectively. The only difference between flight two and flight three were the propellers, the already discussed GS 5.0x4.5 (flight two) and the smaller GS 4.0x4.5 (flight three) were used. The outcome in terms of trajectory was quite similar, as presented in Figure 9, which allows for a fair comparison between propellers. Figure 10 shows that flight number three, with the smaller propeller, shows indeed a slightly smaller current and presents a higher voltage at the end of the flight. However, such improvement would still not make the 23 minutes flight possible as the power demand remains high, as shown in Figure 11. As a result, the autonomous flight capability was validated, and the hypothesis that a bigger propeller was causing an increase in the energy consumption was confirmed, even though the influence is small. Nevertheless, we opted for including

http://www.imavs.org/

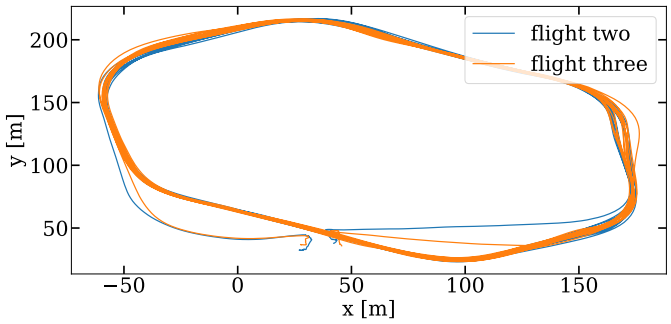


Figure 9: Autonomous flight two and three trajectories in XY plane

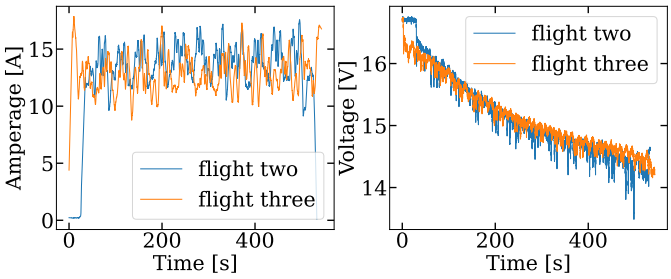


Figure 10: Autonomous flight two and three current and voltage

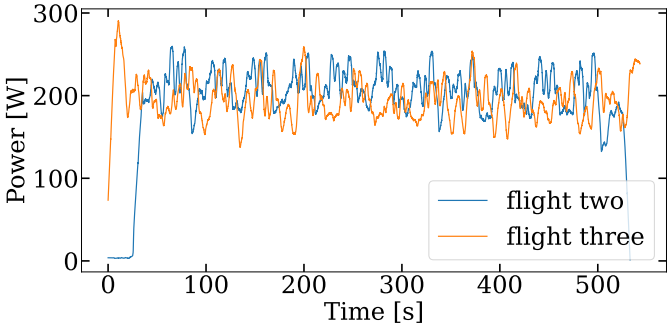


Figure 11: Autonomous flight two and three power consumption

propeller analysis in the second iteration of the MDO, as discussed later.

7.3 Drag evaluation

In order to validate both lift and drag predictions, the vehicle was tested in a wind tunnel. The experiments were performed inside ENAC's (French Civil Aviation University) flight arena in Toulouse, France. The wind tunnel, designed by WindShape [4] has a $1.5\,m$ by $0.75\,m$ open test section and is used both for research and educational purposes. Force and moments were measured with a six-axis ATI Mini-40 sensor [5] calibrated with SI-40-2, with maximum force range of 40N for F_{xy}, 120N for F_z and $1/100\,N$ resolution for xy and $1/50\,N$ resolution for z. The moment range is 2Nm for M_{xyz} with a $1/4000\,Nm$ resolution. The vehicle was tested in a single airspeed representative velocity of 19.5 m/s and for four equally spaced angles of attack between 0°and 22.5°. Figure 12 shows the vehicle in front of the test section.

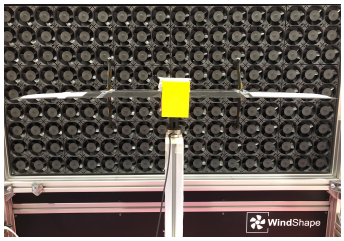


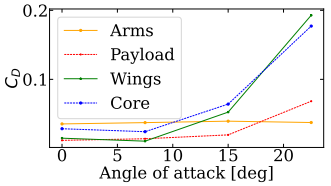
Figure 12: Vehicle attached to the force and moment balance in front of WindShape.

In order to understand the contributions of each drag source in cruise flight, the vehicle was tested in different configurations: full vehicle, without wings, without mounting arms, without wings and mounting arms, and without payload. The result of this test is shown in Figure 13. The core drag is mainly driven by the battery and autopilot module, which are fixed. The test showed that the mounting arms were the most dominant component with respect to drag generation, responsible for almost half of the total drag. Hence we decided to evaluate a different configuration for this component, in order to reduce such behavior. Another vehicle was printed with the arms in "H" configuration, as shown in Figure 14. But little difference was observed. "X" configuration arms led to slightly higher C_L for small angles of attack, as shown in Figure 15. So we chose to maintain the "X" geometry mainly because of its simplicity and light weight.

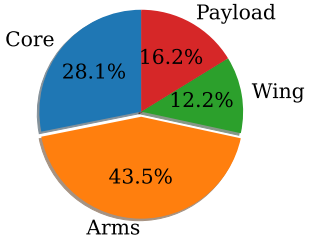
The outcome of the wind tunnel tests was then used to refine the second version of the aerodynamic module of the MDO.

7.4 Lift evaluation

By analyzing Figure 15, it is also possible to observe that the predicted C_L of 0.91 for an angle of attack of about 9°,



(a) Drag with respect to α



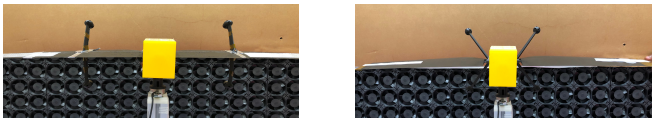
(b) Component contributions at α of 7.5°

Figure 13: Wind tunnel test outcome

[4] https://windshape.com/technology/

[5] https://www.ati-ia.com/products/ft/ft_models.aspx?id=mini40

http://www.imavs.org/



(a) Mounting arms in "H" (b) Mounting arms in "X"

Figure 14: Test setup for arm configuration study. Half of the WindShape test section was covered to ease the visualisation.

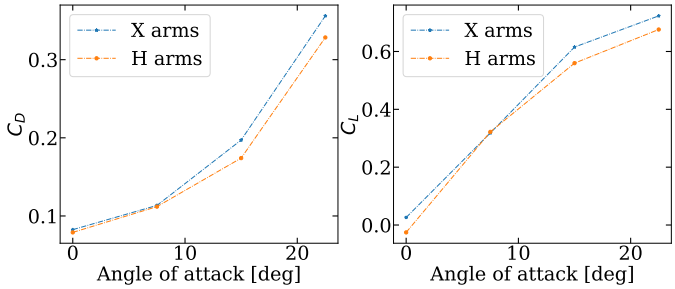


Figure 15: Comparison of arm configuration

shown in Table 7 was unfeasible. The actual value is less than half of the predicted value. The consequence for that can be seen in Figure 16, showing that pitch angle for both flight two and flight three were higher than 20°during the whole cruise flight, a flight regime that can be considered to be stalled, even if the real angle of attack is not known. As the wing lift was overestimated, the autopilot had to increase the pitch angle to ensure that the required force to compensate the weight was generated, but at such high pitch angles, only the motors were generating lifting force, while the stalled wings were mostly contributing to the drag, penalizing flight efficiency.

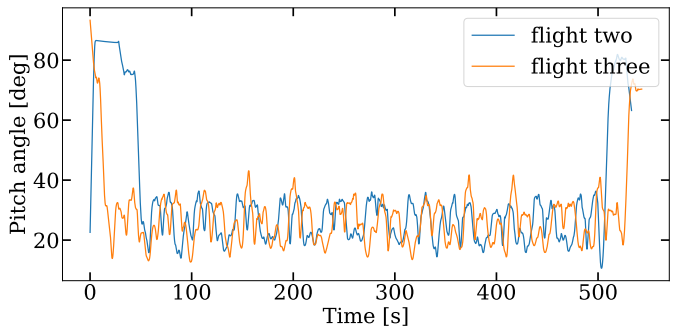


Figure 16: Pitch angle for flight two and three.

This is believed to have caused the lack of performance observed during the initial flights. The lack of lift was most likely generated by non considered interference effects and 3D printing inaccuracies, while the process was still being refined. In order to handle that for the second iteration of the MDO, the lift output from OpenAerostruct was tuned to match the observed data, as it will be shown.

8 MDO Process Refinement

The results obtained in the first batch of flight tests have shown that the initial MDO formulation was possibly over-simplified. The assumption made in Eq. (2) was that, in cruise, thrust is equal to weight, and that the required thrust would be available throughout the entire flight. There were no corrections for lift and drag prediction, which led to a small wing area, incapable of maintaining sustained level flight at low angles of attack and culminating in poor performance. In order to refine our methodology and mitigate these effects, we have added new constraints for hover flight and turning maneuver and refined the aerodynamic module.

8.1 Propulsion refinement

8.1.1 Thrust calculation

In order to obtain thrust (T) in both hover and cruise flight conditions, we employed CCBlade [6], a blade element momentum method formulation that is specially suited for gradient based optimization presented by Ning [22]. As CCBlade is implemented in Julia, we developed a wrapper to call it from our Python environment. With CCBlade, the thrust generated is calculated as a function of airspeed, propeller geometry and rpm. We employed Xfoil [23] to account for the correct propeller airfoil polar in the optimization loop.

8.1.2 Hover analysis

In order to account for the hover capability, a new constraint was implemented:

$$\frac{a_z}{m} = 0.9T - m_{mav}g \geq 0 \tag{8}$$

The acceleration in the z axis (a_z), calculated as the difference between the thrust (T) generated by the four propellers and MAV weight ($m_{mav}g$) is constrained to be greater than 0, in order to ensure hover capability. We added a safety margin of 10% to account for eventually necessary attitude corrections that would require thrust in other directions than z.

8.1.3 Turn maneuver analysis

Similarly, a constraint to ensure that in the forward flight (x) axis there is enough thrust to generate positive or zero acceleration (a_x), was added:

$$\frac{a_x}{m} = T - D \geq 0 \quad at \quad V_{turn} \tag{9}$$

where the both lift ($C_{L_{turn}}$) and drag ($C_{D_{turn}}$) were also obtained using OpenAeroStruct. Thrust (T) is now assumed as the sum of the thrust from all the motors operating with a safety margin of 15%. Velocity (V_{turn}) can be calculated as

$$V_{turn} = \sqrt{\frac{m_{mav}gn}{0.5\rho S C_{L_{turn}}}} \tag{10}$$

[6] https://github.com/byuflowlab/CCBlade.jl

http://www.imavs.org/

where the load factor n accounts for the decrease in lift generated by the bank angle.

Considering this more detailed problem formulation, we were also able to refine range and endurance calculation. The turning radius from Eq. (5) is now calculated as a function of V_{turn} :

$$T_R = \frac{V_{turn}^2}{g\sqrt{n^2-1}}. \tag{11}$$

Such new formulation allowed to split each lap into two different flight conditions: 1000 m cruise flight and a bank flight with a length of $2\pi T_R$ m. With that, relations for average airspeed, lift and drag coefficient were adopted:

$$V_{avg} = V_{cruise}(1 - \frac{2\pi T_R}{1000}) + V_{turn}\frac{2\pi T_R}{1000} \tag{12}$$

$$C_{L_{avg}} = C_{L_{cruise}}(1 - \frac{2\pi T_R}{1000}) + C_{L_{turn}}\frac{2\pi T_R}{1000} \tag{13}$$

$$C_{D_{avg}} = C_{D_{cruise}}(1 - \frac{2\pi T_R}{1000}) + C_{D_{turn}}\frac{2\pi T_R}{1000} \tag{14}$$

which in turn allow us to refine the range from Eq. (3) and endurance from Eq.(4):

$$R = \frac{m_{bat}}{m_{mav}g}B_{SE}\eta\frac{C_{L_{avg}}}{C_{D_{avg}}} \qquad E = \frac{R}{V_{avg}} \tag{15}$$

8.1.4 Aerodynamic calculation refinement

In order to enhance the drag prediction in the MDO process, we used the wind tunnel test results at representative velocity to refine the drag calculation. As shown in Figure 13a, both payload and motor arm drag can be assumed as constant at such speed for angles of attack lower than 15°. The core drag was approximated with a second order polynomial function, varying according to the angle of attack. The wing drag, calculated from OpenAeroStruct, was kept without changes. The total drag was then calculated as:

$$C_D = C_{D_{OAS}} + C_{D_{core}}(\alpha) + C_{D_{constant}} \tag{16}$$

where

$$C_{D_{constant}} = C_{D_{arms}} + C_{D_{Payload}} \tag{17}$$

For the lift, two changes were made: The predicted lift coefficient for alpha zero (C_{L0}) was updated from 0.2 (first MDO iteration) to zero, and a correction of 0.7 was also applied to the output of the OpenAeroStruct. These changes were made with the objective of fitting the predicted lift to the observed one.

8.1.5 Weights module refinement

The weight prediction of the components was also updated considering the information gathered with the first vehicle, as shown in Table 8.

Component	Mass
Motors and arms	104 $[g]$
Cables	20.0 $[g]$
Wing density of area	1360 $[g/m^2]$

Table 8: Fixed weight breakdown

8.2 Refined MDO formulation

Table 9 shows the new configuration of the MDO problem, with the added design variable and constraints. The design space for the second iteration had to be reduced in order to comply with the manufacturing technique limitation, as problems occurred when printing larger wing spans and smaller tip chords. As the wind tunnel campaign was fully conducted considering the initially defined payload of 0.2 kg, this value was fixed in order to ensure precise drag prediction and avoid extrapolations with untested payloads. Even if the analysis presented in Section 4 could have been refined considering the new data, we considered that the trends observed with the first MDO problem were still valid.

	Function/Variable	Lower	Upper
Maximize	score		
With respect to	Root chord	0.125 m	0.2 m
	Tip chord	0.095 m	0.2 m
	Span	0.3 m	0.7 m
	Cruise α	1°	10°
	Turn α	1°	8°
	Battery mass	0.1 kg	0.2 kg
Subject to	Endurance		23 min
	a_x	0	
	a_x	0	

Table 9: MDO problem formulation

For this execution we used the super efficient global optimization coupled with mixture of experts (SEGOMOE) [24], a surrogate-based gradient-free optimizer that can handle expensive and multimodal cases, in order to avoid converging to a local optima.

9 OPTIMIZED VEHICLE AND FLIGHT TEST

Table 10 shows the characteristics of the second iteration of the vehicle.

The optimal geometry was then manufactured and flight tested. The vehicle weight was 753 g, 13 g higher than the predicted with the MDO. Due to problems with the guidance module, the flight with the new geometry was not autonomous, what entangles the discussion on flight efficiency. Nevertheless, the flight trajectory was uniform, as shown in Figure 17.

The pitch angle, shown in Figure 18, was smaller and mostly below stall region, which in turn allowed for the de-

http://www.imavs.org/

Design variables	Value
Root chord	0.12 m
Tip chord	0.095 m
Wing span	0.7 m
Cruise α	10°
Turn α	8°
Battery mass	0.2 kg
Coupling variables	**Value**
Wing area	0.075 m^2
Cruise speed	18.0 m/s
Cruise C_L	0.54
Cruise C_D	0.12
Flight time	14.6min
Number of laps	13
MAV mass	0.74 kg

Table 10: Optimal MAV characteristics at the end of new MDO iteration

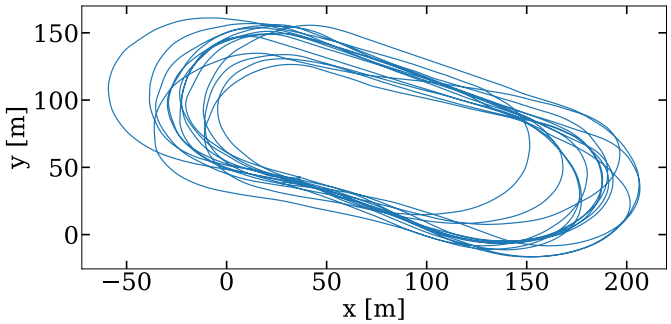


Figure 17: 4th flight XY trajectory.

sired reduction in power consumption, shown in Figure 19.

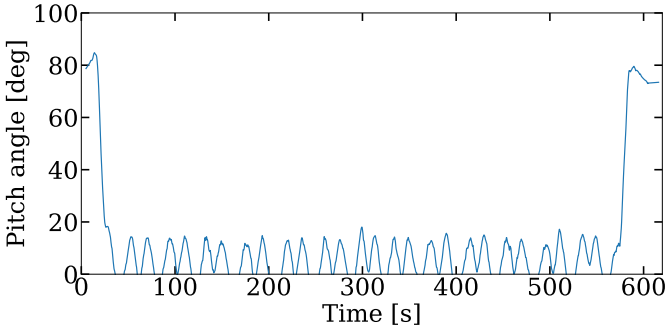


Figure 18: Pitch angle variation during the 4th flight.

During this approximately 10 minutes of flight with longer than usual hover phases, conducted to re-identify the vehicle and refine the INDI controller coefficients, the energy consumed was roughly 19.4Wh, using a 4s LiPo battery with 26.9Wh nominal capacity. Considering the flight data, it is estimated that, with the same battery and assuming a complete run consuming around 23Wh to ensure a safety

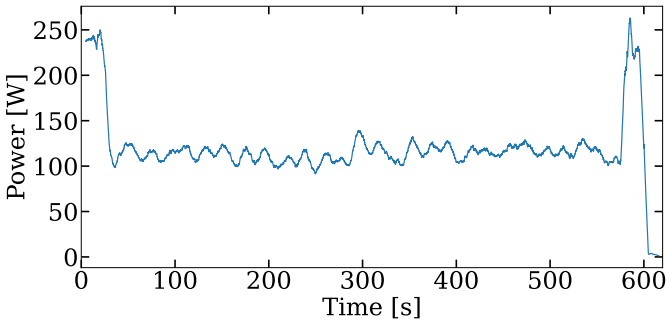


Figure 19: Power consumption during the 4th flight

margin, the predicted flight duration of 14 minutes can be achieved, specially taking into account the benefits from the autonomous flight and a smaller hover phase, which is the most energy consuming portion of the flight.

10 Conclusion

In this paper we presented a full design and flight testing cycle of an MAV. The vehicle was designed to comply with the IMAV 2022 competition rules, an interesting benchmark for evaluating a design methodology. A multidisciplinary design and optimization approach was selected. The first MDO was a combination of three disciplines: aerodynamics, weights, and mission analysis, and it led to a smaller design, which was built and flight tested. Driven by the necessity of reducing power consumption observed in the first flight test, we conducted wind tunnel tests to identify the main sources of drag and refined the MDO process by adding an improved lift and drag prediction module, created with the results of the experiments, and CCBlade for thrust analysis in both hover and turn maneuver conditions. The design obtained after the second iteration of the MDO presented a more accurate lift prediction, in line with the expectation, which in turn resulted in lower energy consumption during test flights. For future work and the IMAV competition, studies addressing core and payload drag reduction will be conducted, "Li-Ion" batteries will be considered, as existing batteries with 37Wh nominal capacity could allow even longer flights, and the manufacturing process will be refined with the objective of allowing for higher wing spans and, as a result, more design freedom. Additionally, the outcome of autonomous and competition flights are intended to be published in a journal version of this work.

Acknowledgments

This work is part of the activities of ONERA - ISAE - ENAC joint research group.

References

[1] Elena Ausonio, Patrizia Bagnerini, and Marco Ghio. Drone swarms in fire suppression activities: A conceptual framework. *Drones*, 5(1), 2021.

http://www.imavs.org/

[2] Moulay A. Akhloufi, Andy Couturier, and Nicolás A. Castro. Unmanned aerial vehicles for wildland fires: Sensing, perception, cooperation and assistance. *Drones*, 5(1), 2021.

[3] Yuri Taddia, Corinne Corbau, Joana Buoninsegni, Umberto Simeoni, and Alberto Pellegrinelli. Uav approach for detecting plastic marine debris on the beach: A case study in the po river delta (italy). *Drones*, 5(4), 2021.

[4] Ágoston Restás. Drone applications fighting covid-19 pandemic;towards good practices. *Drones*, 6(1), 2022.

[5] Zichao Zhang and Davide Scaramuzza. Perception-aware receding horizon navigation for mavs. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, page 2534–2541. IEEE Press, 2018.

[6] Kirk Y.W. Scheper and Guido C.H.E. de Croon. Evolution of robust high speed optical-flow-based landing for autonomous mavs. *Robotics and Autonomous Systems*, 124:103380, 2020.

[7] Robin Ritz and Raffaello D'Andrea. A global controller for flying wing tailsitter vehicles. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2731–2738, 2017.

[8] Philipp Hartmann, Carsten Meyer, and Dieter Moormann. Unified velocity control and flight state transition of unmanned tilt-wing aircraft. *Journal of Guidance, Control, and Dynamics*, 40(6):1348–1359, 2017.

[9] Ewoud J. J. Smeur, Murat Bronz, and Guido C. H. E. de Croon. Incremental control and guidance of hybrid aircraft applied to a tailsitter unmanned air vehicle. *Journal of Guidance, Control, and Dynamics*, 43(2):274–287, 2020.

[10] Bo Wang, Zhongxi Hou, Zhaowei Liu, Qingyang Chen, and Xiongfeng Zhu. Preliminary design of a small unmanned battery powered tailsitter. *International Journal of Aerospace Engineering*, 2016:3570581, May 2016.

[11] J. Holsten, T. Ostermann, and D. Moormann. Design and wind tunnel tests of a tiltwing uav. *CEAS Aeronautical Journal*, 2(1):69–79, Dec 2011.

[12] Tomáš Vogeltanz. Conceptual design and control of twin-propeller tail-sitter mini-uav. *CEAS Aeronautical Journal*, 10(3):937–954, Sep 2019.

[13] 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.

[14] Andrew B. Lambe and Joaquim R. R. A. Martins. Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes. *Structural and Multidisciplinary Optimization*, 46:273–284, 2012.

[15] Rémi Lafage, Sebastien Defoort, and Thierry Lefebvre. Whatsopt: a web application for multidisciplinary design analysis and optimization. In *AIAA Aviation 2019 Forum*, page 2990, 2019.

[16] John P. Jasa, John T. Hwang, and Joaquim R. R. A. Martins. Open-source coupled aerostructural optimization using Python. *Structural and Multidisciplinary Optimization*, 57(4):1815–1827, April 2018.

[17] J. Carvill. 4 - fluid mechanics. In J. Carvill, editor, *Mechanical Engineer's Data Handbook*, pages 146–171. Butterworth-Heinemann, Oxford, 1993.

[18] Murat Bronz, Jacson Barth, Ezra Tal, Federico Favalli, and Sertac Karaman. Mission-oriented additive manufacturing of modular mini-uavs. In *AIAA Scitech 2020 Forum*, 2020.

[19] Justin S. Gray, John T. Hwang, Joaquim R. R. A. Martins, Kenneth T. Moore, and Bret A. Naylor. OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization. *Structural and Multidisciplinary Optimization*, 59(4):1075–1104, April 2019.

[20] Neil Wu, Gaetan Kenway, Charles A. Mader, John Jasa, and Joaquim R. R. A. Martins. pyoptsparse: A python framework for large-scale constrained nonlinear optimization of sparse systems. *Journal of Open Source Software*, 5(54):2564, 2020.

[21] Pascal Brisset, Antoine Drouin, Michel Gorraz, Pierre-Selim Huard, and Jeremy Tyler. The paparazzi solution. In *IMAV 2006*, 10 2006.

[22] Andrew Ning. Using blade element momentum methods with gradient-based design optimization. *Structural and Multidisciplinary Optimization*, 64(2):991–1014, August 2021.

[23] Mark Drela. Xfoil: An analysis and design system for low reynolds number airfoils. In Thomas J. Mueller, editor, *Low Reynolds Number Aerodynamics*, pages 1–12, Berlin, Heidelberg, 1989. Springer Berlin Heidelberg.

[24] N. Bartoli, T. Lefebvre, S. Dubreuil, R. Olivanti, R. Priem, N. Bons, Joaquim R. R. A. Martins, and J. Morlier. Adaptive modeling strategy for constrained global optimization with application to aerodynamic wing design. *Aerospace Science and Technology*, 90:85–102, July 2019.

http://www.imavs.org/

C wing geometric redesign based on aircraft flying qualities

María Beatriz Bernábe Loranca[1]*, and H. Rodríguez-Cortés[2]†
[1] Facultad de Ciencias de la Computación, Benemérita Universidad Autónoma de Puebla, Puebla, México.
[2] Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional,
Av. Instituto Politpecnico Nacional 2508, San Pedro Zacatenco, 07360, México, D.F., México

http://www.imavs.org/

ABSTRACT

This work formulates the application of a multi-objective design methodology for the conceptual design of a C-wing tip for a flying wing powered by an electric power plant. The design objective is to propose C-wing geometric configurations regarding the longitudinal flying qualities. The optimization objectives consider the short-period mode frequency and the time it would take to duplicate the amplitude of the oscillations after being disturbed. A Multi-objective evolutionary algorithm (MOEA) that uses the so-called meta-heuristic Differential Evolution solves the multi-objective optimization problem. The selection of the solutions representing the best compromise between the optimization goals is based on Pareto dominance.

1 INTRODUCTION

Advances in various technological areas have fostered industry and research centers to search for new solutions or improvements to existing ones constantly. In most cases, the goal of innovation can be translated into an optimization problem. In particular, in Aeronautical engineering, aircraft design is, by nature, multi-objective and multidisciplinary, with several conflicting design objectives. Motivated by economic and industrial requirements, aircraft design has experienced innovative changes and, during the last decades, has been considerably improved by using computer simulations [17, 18, 2]. For example, computational fluid dynamics (CFD) for aerodynamic analysis [13, 14, 20, 19], and finite element method (FEM) for structural analysis [15] and [11, 29]. The continuous demand for better designs, subject to economic and environmental constraints, and the continuous development in computer simulation techniques have led to highly computerized aircraft design processes. Unmanned Aerial Vehicles (UAVs) of any size have proven effective in various applications, raising great interest in developing new configurations. Configurations are limited only by the design team's imagination during the conceptual design phase. This high degree of freedom makes the decision-making process complicated. Thus, optimization techniques are of high value in these design stages. In particular, multi-objective optimization techniques are preferred as they provide a set of solutions instead of a single solution. The set of solutions represents the best compromise between the different design objectives such that the decision-making can be quantified.

The flying qualities of human-crewed aircraft have been studied for a long time based on the pilot's evaluation [31]. Their study has achieved a high degree of maturity. It is known with precision the qualities an aircraft must satisfy based on its mission profile, and there is an aircraft classification regarding flying qualities [5]. UAVs' flight qualities must be radically different from the definition of the flying qualities for full-scale aircraft since UAVs are pilotless. Some efforts were made in the seventies to determine flying qualities for remotely piloted aircraft. These efforts tried to define the flying qualities of UAVs, simply scaling the flying qualities of a complete-scale aircraft [24]. Recent efforts to define UAV flying qualities have been reported in [7, 4, 27, 16].

This paper addresses the conceptual redesign of a C-wing tip geometry for a small flying wing powered by an electric power plant using a multi-objective optimization methodology. The design explores the relationship between the C-wing geometric characteristics and the resulting aircraft dynamic characteristics expressed in the flying qualities. The design objective is stated as an optimization problem in terms of the short-period mode frequency and the time it would take to duplicate the amplitude of the oscillations after disturbances. A Multi-objective evolutionary algorithm based on the meta-heuristic differential evolution solves the optimization problem. The selection of the solutions that represent the best compromise between the optimization goals is based on Pareto dominance.

This paper has the following structure. Section 2 describes the multi-objective optimization procedure and presents the study's case. Section 3 introduces the objective functions, establishes the optimization problem, and describes the solution algorithm. Section 4 reports the results obtained from the optimization algorithm. Finally, Section 5 reports concluding remarks.

*Email address(es): beatriz.bernabe@gmail.com
†Email address(es): hrodriguez@cinvestav.mx

http://www.imavs.org/

2 MULTI-OBJECTIVE OPTIMIZATION

A multi-objective optimization problem can be defined mathematically as

$$\text{minimize } \vec{f}(\vec{x}) := [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})] \tag{1}$$

constrained to:

$$g_i(\vec{x}) \leq 0 \quad i = 1, 2, \ldots, m \tag{2}$$

$$h_i(\vec{x}) = 0 \quad i = 1, 2, \ldots, p \tag{3}$$

where $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ is the decision variables vector, which are bounded by lower limits (x_i^l) and upper limits (x_i^u) defining the search space $\mathcal{S}$. $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., k$ are the objective functions and $g_i, h_j : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., m$, $j = 1, ..., p$ are the constraint functions. A minimization problem is assumed to be addressed without loss of generality.

A multi objective problem aims to determine from a set of possible solutions $\mathcal{F}$, satisfying the constraints, those corresponding to optimal values for all k objective functions simultaneously [9]. Any decision variable vector $\vec{x}$ that satisfies the constraints is considered a feasible solution. Pareto dominance is an important ingredient for the notion of optimality in multi- objective optimization problems. To define the Pareto dominance the following definitions are introduced.

Definition 1 *A decision variable vector $\vec{x}$ dominates a second one $\vec{y}$, denoted $\vec{x} \preceq \vec{y}$, if and only if $\vec{x}$ is partially less than $\vec{y}$, in the following sense*

$$\forall\, i \in \{1, \ldots, k\},\ f_i(\vec{x}) \leq f_i(\vec{y}) \wedge \exists\, i \in \{1, \ldots, k\} \ :\ f_i(\vec{x}) < f_i(\vec{y}).$$

Definition 2 *A decision variable vector $\vec{x} \in \mathcal{S} \subset \mathbb{R}^n$ is* **not***-dominated with respect to $\mathcal{S}$, if there is no other vector $\vec{x}' \in \mathcal{S}$ such that $\vec{f}(\vec{x}') \preceq \vec{f}(\vec{x})$.*

To ensure that a solution **dominates** another, it is necessary that the dominating solution is strictly better in at least one design objective and not worse in any of the other design objectives. Thus, for a multi-objective optimization problem, when a comparison between two different solutions $\vec{x}$ and $\vec{y}$ is made, the following possibilities arise [12], [3].

- $\vec{x} \preceq \vec{y}$, $\vec{x}$ dominates $\vec{y}$. $\vec{x} \succeq \vec{y}$, $\vec{x}$ is dominated by $\vec{y}$.
- $\vec{x} \not\preceq \vec{y} \wedge \vec{y} \not\preceq \vec{x}$, $\vec{x}$ and $\vec{y}$ are not comparable, it is not possible to establish which solution dominates.

Definition 3 *A decision variable vector $\vec{x}^* \in \mathcal{F} \subseteq \mathcal{S}$ is a* **Pareto Optimal** *if it is non-dominated with respect to $\mathcal{F}$.*

A Pareto optimum is a solution that cannot be improved concerning an objective function without worsening other. Hence, a Pareto optimum does not provide a unique solution for the space of the decision variables but provides a set of solutions that form the so-called *Pareto Optimal Set* $\mathcal{P}^*$.

Definition 4 *The* **Pareto Optimal Set** $\mathcal{P}^*$ *is determined as*

$$\mathcal{P}^* = \{\vec{x} \in \mathcal{F} \mid \vec{x} \text{ is a Pareto Optimal}\}$$

The solutions contained in $\mathcal{P}^*$ are non-dominated *between them.*

All non-dominated solutions plotted in the objective function space trace the Pareto Optimal called *Pareto front* $\mathcal{PF}^*$.

Definition 5 *The* **Pareto front** $\mathcal{PF}^*$ *is described as*

$$\mathcal{PF}^* = \{\vec{f}(\vec{x}) \in \mathbb{R}^k \mid \vec{x} \in \mathcal{P}^*\}$$

The goal in solving multi-objective problems is to determine $\mathcal{P}^*$. Hence, in solving a multi-objective problem, there is no interest in finding a unique solution but the set of solutions representing the best possible compromise between the problem's objectives. Because, in general, it is impossible to find an analytical expression that defines the Pareto front of a multi-objective problem, the commonly used approach is to compute a sufficient number of points in the feasible region, which are then filtered based on the Pareto dominance, to thereby obtaining an approximation of the Pareto front shape.

2.1 The case of study

The aerial platform of this work is a flying wing equipped with a C-wing tip, as shown in Figure 1. This aircraft configuration is derived from [10], where a flying wing with a conventional winglet was designed for aerodynamic parameter identification purposes. The resulting flying wing was unstable because reflex aerodynamic profiles were not used, hindering its radio control human-crewed operation. Here, it is proposed to use a C-wing tip to solve this problem. The C-wing dimensioning problem to reduce aerodynamic drag under different optimization criteria has been addressed in [23, 26, 22]. In [23], the C-wing dimensions were optimized to reduce the aerodynamic drag constrained to generate a statically stable pitching moment. The C-wing tip may help to change the slope of the pitching moment coefficient modifying the aircraft stability properties [23]; consequently, the difficulties in flying the aircraft may be reduced. However, the next question arises: how much this slope needs to be modified? This work proposes a partial answer to that question by redesigning the C-wing geometric dimensions in terms of the longitudinal aircraft flying qualities; this is the main innovation of this work.

3 OPTIMIZATION PROBLEM

The role of the objective functions is to link the C-wing tip geometry with the dynamic aircraft characteristics. Since the aircraft aerodynamic coefficients depend on the aircraft geometry and the dynamic aircraft characteristics depend on the aerodynamic coefficients, the aircraft longitudinal dynamics is an excellent candidate to link all design variables. The

CS chord	0.3m	Wing root chord	0.3m
Wingtip chord	0.15m	Winglet tip chord	0.1m
C tip chord	0.0579m	CS ap	NACA0018
Wing section ap	Eppler 66	C wing ap	NACA 0009
Winglet ap	NACA 009	i_{CS}	0^o
i_{WS}	0^o	i_{wn}	0^o
i_C	-10^o	b_{CS}	0.25m
b_{WS}	0.8m	b_C	0.2215m
Λ_{WS}	52.7515^o	Λ_{wn}	32.5785^o
Λ_C	-29.0231^o	h_{wn}	0.15m

Table 1: Geometric characteristics of the flying wing base design. i incidence, Λ sweep, b span, ap aerodynamic profile. CS central section, WS wing section, wn winglet, C C-wing, h height.



Figure 1: Geometry of the UAV with a C wing tip.

aircraft longitudinal dynamics is described by the following equations [25]

$$\begin{aligned} m\dot{V} &= T\cos(\alpha) - \frac{1}{2}\rho V^2 S C_D(\alpha) - mg\sin(\gamma), \\ mV\dot{\gamma} &= -T\sin(\alpha) + \frac{1}{2}\rho V^2 S C_L(\alpha, q, \delta_e) - mg\cos(\gamma) \\ \dot{\theta} &= q, \\ I_{yy}\dot{q} &= \frac{1}{2}\rho V^2 S c C_M(\alpha, q, \delta_e) + T\ell_z \end{aligned} \tag{4}$$

where T is the power plant thrust, $\alpha = \theta - \gamma$ is the angle of attack, V is the aircraft airspeed, m is the aircraft mass, γ is the angle of flight, θ is the pitch angle, q is the pitch rotational speed, δ_e is the angle of deflection of the elevator, C_D, C_L and C_M are the aerodynamic coefficients drag, lift and pitching moment, respectively. Finally, ρ is the air density, S is the wing area, g is the gravity acceleration constant, I_{yy} is the inertia moment around the lateral axis, c is the wing mean aerodynamic chord, and ℓ_z is the vertical distance from the longitudinal axis of the power plant and the aircraft center of gravity.

It is assumed that the aerodynamic coefficients have the following structure [28]

$$\begin{aligned} C_D(\alpha) &= C_{D_0} + C_{D_1}\alpha + C_{D_2}\alpha^2, \\ C_L(\alpha, q, \delta_e) &= C_{L_0} + C_{L_1}\alpha + C_{L_q}\frac{qc}{2V} + C_{L_e}\delta_e \\ C_M(\alpha, q, \delta_e) &= C_{M_0} + C_{M_1}\alpha + C_{M_q}\frac{qc}{2V} + C_{M_e}\delta_e \end{aligned} \tag{5}$$

with $C_{D_0}, C_{D_1}, C_{D_2}, C_{L_0}, C_{L_1}, C_{L_q}, C_{L_e}, C_{M_0}, C_{M_1}, C_{M_q}$ and C_{M_e} are coefficients obtained from the CMARC software. Cmarc is a inviscid fluid flow analysis program of the type known as a low-order panel method. In order to obtain the aircraft dynamic modes it is necessary to linearize the nonlinear model in (4) around an equilibrium (trim) point [28]. The equilibrium point of (4) is defined by $\bar{V}$, $\bar{\alpha} = \bar{\theta} - \bar{\gamma}$, $\bar{\theta}$, $\bar{q}$, $\bar{T}$ and $\bar{\delta}_e$ such that

$$\begin{aligned} 0 &= \bar{T}\cos(\bar{\alpha}) - \frac{1}{2}\rho\bar{V}^2 S C_D(\bar{\alpha}) - mg\sin(\bar{\gamma}), \\ 0 &= -\bar{T}\sin(\bar{\alpha}) + \frac{1}{2}\rho\bar{V}^2 S C_L(\bar{\alpha}, \bar{q}, \bar{\delta}_e) - mg\cos(\bar{\gamma}) \\ 0 &= \bar{q}, \\ 0 &= \frac{1}{2}\rho\bar{V}^2 S c C_M(\bar{\alpha}, \bar{q}, \bar{\delta}_e) + \bar{T}\ell_z \end{aligned} \tag{6}$$

To obtain an equilibrium point, the airspeed and the flight path angle are fixed at $\bar{V} = 15$m/s, $\bar{\gamma} = 0$ rad, respectively. Hence, there are three equations and three unknown variables $\bar{T}$, $\bar{\alpha}$ and $\bar{\delta}_e$. Defining

$$x = \begin{bmatrix} V & \gamma & \theta & q \end{bmatrix}^\top, \; u = \begin{bmatrix} T & \delta_e \end{bmatrix}^\top$$

the aircraft longitudinal dynamics (4) can be written in compact form as follows

$$\dot{x} = F(x, u) \tag{7}$$

The linearized model of (7) around the equilibrium point

$$(\bar{x}, \bar{u}) = \{\bar{x}, \bar{u} \mid F(\bar{x}, \bar{u}) = 0\}$$

is given by

$$\dot{x}_\delta = Ax_\delta + Bu_\delta,$$

with

$$A = \frac{\partial F}{\partial x}\Big|_{x=\bar{x}, u=\bar{u}}, \; B = \frac{\partial F}{\partial u}\Big|_{x=\bar{x}, u=\bar{u}}$$

and (x_δ, u_δ) the approximation of (x, u) in the neighborhood of $(\bar{x}, \bar{u})$. The aircraft modes of motion are determined by the eigenvalues of the matrix A. The matrix A summarizes the aircraft dynamics characteristics with the control inputs fixed at the value corresponding to the equilibrium point. The eigenvalues have the following form [28]

$$\lambda_{1,2} := -a \pm bi, \; \lambda_{3,4} := -c \pm di$$

from where the frequencies and the damping coefficients of the modes of motion can be obtained as follows

$$\begin{aligned} \omega_{n_1} &= \sqrt{a^2 + b^2}, \; \zeta_{n_1} = \frac{a}{\omega_{n_1}}, \\ \omega_{n_2} &= \sqrt{c^2 + d^2}, \; \zeta_{n_2} = \frac{c}{\omega_{n_2}} \end{aligned} \tag{8}$$

If $\omega_{n_1} > \omega_{n_2}$ then ω_{n_1} defines the frequency of the short- period and ω_{n_2} is the frequency of the long-period (phugoide). The short and long period values as well as the corresponding

http://www.imavs.org/

damping coefficients can be used to characterize the aircraft flying qualities as proposed in [5].
In this framework, the optimization problem is defined as follows. Let

$$\vec{x} \quad = \quad \{h_{wn}, \Lambda_{wn}, i_C, b_C, \Lambda_C\}$$

be the vector of decision variables. Consider the following optimization objectives

$$f_1(\vec{x}) = \max(\omega_{n_1}, \omega_{n_2}), \quad f_2(\vec{x}) = -T_{2ph} \tag{9}$$

where

$$T_{2ph} = \frac{\ln(2)}{-\zeta_{ph}\omega_{nph}} \tag{10}$$

with ζ_{ph} the damping coefficient for the phugoid or long-period oscillation mode. The value of T_{2ph} characterizes the time it would take for an unstable aircraft to double the amplitude of the oscillations after being disturbed. For unstable aircraft, this value is positive. For a stable aircraft, the value of T_{2ph} is negative, and its numerical value is the time it takes the aircraft to halve the oscillation amplitude after a disturbance. The proposed design objectives try to find a compromise between the two motion modes of an aircraft.
Now, we are in position to establish the optimization problem which in this case takes the following form

$$\min \{f_1(\vec{x}), f_2(\vec{x})\} \tag{11}$$

constraint to

$$\begin{array}{ll} 0.1m \leq h_{wn} \leq 0.2m, & 0.0^o \leq \Lambda_{wn} \leq 45^o, \\ -10^o \leq i_C \leq +5^o, & 0.1m \leq b_C \leq 0.25m, \\ -30^o \leq \Lambda_C \leq +30^o & \end{array} \tag{12}$$

Once we have defined the objective functions to optimize (minimize, in this case), we must implement a multi-objective optimization method. In the literature, various optimization methods range from those using mathematical programming techniques [21] to those using evolutive meta-heuristic algorithms [6]. For the latter case, we can distinguish algorithms based on genetic algorithms [8], evolutionary strategies [33], clusters of particles in [30], Differential Evolution [32] , among others. This article uses a multi-objective evolutionary algorithm, called MODE -LD + SS, proposed in [1]. This algorithm makes use of meta-heuristics called Differential Evolution. The multi-objective optimization process will start with a population of 50 individuals. Since Evolutionary Algorithms are stochastic processes, several runs are needed for several generations. In these runs, possible solutions were accumulated in the order of a thousand. The solutions found from different runs were processed to determine those that are globally non-dominated and thus approximate the Pareto front solution representing the problem. The aerodynamic properties of all tested geometric configurations are obtained during the optimization process using the CFD program CMARC. The following pseudocode illustrates the workflow.

Algorithm 1 Workflow

Require: 50 aircraft configurations.

Ensure: Design variables on (12).

1: **for** $i = 50, ..., 1000$ **do**
2: Aerodynamic coeffients determination using CMARC.
3: Objective functions computation.
4: **end for**
5: Search for non-dominated solutions.

Figure 2 shows the flying wing representation to be used in the CMARC software.

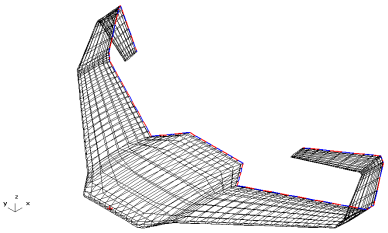


Figure 2: Aircraft panels model to be used by CMARC

4 RESULTS

This section presents the results obtained from the multi-objective optimization process. Figure 3 shows the approximation of the Pareto front obtained for this problem; the front has approximately 52 non-dominated solutions. From

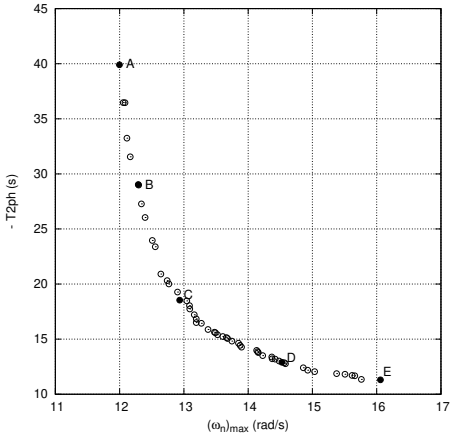


Figure 3: Multiple solutions at the Pareto front.

the Pareto front, it can be observed that there is indeed a con-

http://www.imavs.org/

http://www.imavs.org/

flict between the two objectives considered in this problem. Additionally, it can be verified that all obtained aircraft configurations are stable. Each solution represents a combination of different short and long modes. Five solutions are identified in the Pareto front of Figure 3, labeled A, B, C, D, and E, respectively. Solution A has the minimum value of the first objective function and the maximal value for the second objective function. Thus, the corresponding aircraft will have a lightly damped long-period mode and a minor short-period frequency. The solution E has antisymmetric characteristics, a better damped long-period, and a more significant short-period frequency. Solution C is in the middle part of the Pareto front approximation, and it could be considered the one that represents the best compromise between the two objective functions. In order to compare the different sets of geometric parameters obtained in each of the five selected solutions, Table 2 shows the respective values for the geometric variables. The geometry of the selected solutions is shown in figure 4. From the analysis of the results, as well as the tabulated geometric characteristics for the five selected configurations, it can be observed that the parameters defined as variables in the optimization problem significantly influence the objective functions. Consequently, this influence reflects on the flying qualities of the UAV platform, defined in terms of the frequencies of the aircraft's motion modes.

Solution	h_{wn}	Λ_{wn}	i_C	b_C	Λ_C
A	0.1000	37.6814	3.0417	0.1000	20.9189
B	0.1336	32.8245	4.9212	0.1000	26.6503
C	0.1220	45.0000	5.0000	0.1423	-13.2555
D	0.1026	0.0000	5.0000	0.2500	-30.0000
E	0.1870	0.0000	4.3187	0.2408	-27.5345

Table 2: Design variables for the solutions A,B,C,D, and E on the Pareto front

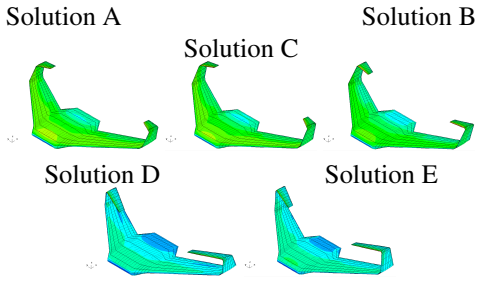


Figure 4: Selected solutions from the Pareto front.

It is difficult to identify trends in the design variables, such as the height of the winglet h_{wn} and the incidence of the C-wing tip i_C, because it is the combination and interaction with other variables that generate the aircraft's flying qualities. Some characteristics of the solutions can be described as follows. Note that winglet swept Λ_{wn} goes towards the upper limit in the solution at the central part of the Pareto front, solution C. For the C-wing swept Λ_C, it is observed that positive values are associated with solutions towards the left part of the Pareto front, solutions A and B, while in the central and right part of the front, the solutions take a negative values. Moreover, small values for the C-wing span b_C are associated with solutions on the left side and up to the central part of the Pareto front, while the solutions on the right side take large C-wing span values near the upper limit.
Concerning the compromise between the objective functions, it is observed that the solutions with the lowest value of the short period consequently have a more extended long period, reaching times of up to 40 seconds to damp to half the oscillation after a disturbance. The multi-objective analysis allows designers to identify solutions and to identify for each of them the values of the objective functions. From the solution set, the control system designer will decide which would be the best choice regarding the bandwidth of the sensors, actuators, and the processing device installed in the UAV control system. Since all aircraft configurations are stable, solution C will be preferred by the control system designer, while solution A will be preferred by a human pilot.

5 CONCLUSIONS

This work presents the application of a multi-objective design methodology to the practical case of the conceptual design of a C-wing tip for a flying wing UAV platform. Five design variables are considered to explore the relationship between the aircraft's geometric characteristics and its dynamic characteristics expressed in terms of flying qualities expressed as the modes of longitudinal flight dynamics. The selection of the solutions that represent the best compromise between the different objectives was based on Pareto dominance. Since it is difficult to identify trends in design variables, such as the height of the winglet h_{wn} and the incidence of the C-wing tip i_C, because it is the combination and interaction with other variables that generate the aircraft flying qualities, it is proposed as future work to develop a factorial statistical experiment design to calibrate the design variables. Consequently, the design of experiments will provide information to establish the most precise optimization area through response surfaces.

REFERENCES

[1] Arias Montaño, A., Coello Coello, C.A., Mezura-Montes, E.: MODE-LD+SS: A Novel Differential Evolution Algorithm Incorporating Local Dominance and Scalar Selection Mechanisms for Multi-Objective Optimization. In: 2010 IEEE Congress on Evolutionary Computation (CEC'2010). IEEE Press, Barcelona, Spain (July 2010)

[2] Benaouali, A., Kachel, S.: Multidisciplinary design optimization of aircraft wing using commercial software integration. Aerospace Science and Technology **92**, 766–776 (2019). https://doi.org/https://doi.org/10.1016/j.ast.2019.06.040, https://www.sciencedirect.com/science/article/pii/S1270963818324131

[3] Bernábe-Loranca, B., Coello-Coello, C., Osorio-Lama, M.: A multi-objective approach for the heuristic optimization of compactness and homogeneity in the optimal zoning. Journal of applied research and technology **10**(3), 447–457 (2012)

[4] Capello, E., Guglieri, G., Marguerettaz, P., Quagliotti, F.: Preliminary assessment of flying and handling qualities for mini-uavs. Journal of Intelligent & Robotic Systems **65**(1), 43–61 (2012)

[5] Chalk, C.R., Neal, T., Harris, T., Pritchard, F.E., Woodcock, R.J.: Background information and user guide for mil-f-8785b (asg),'military specification-flying qualities of piloted airplanes' (1969)

[6] Coello Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems. Springer, New York, second edn. (September 2007), iSBN 978-0-387-33254-3

[7] Cotting, M.C.: Evolution of Flying Qualities Analysis: Problems for a New Generation of Aircraft. Ph.D. thesis, Virginia Polytechnic Institute and State University (2010)

[8] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (April 2002)

[9] Ehrgott, M.: Multicriteria optimization, vol. 491. Springer Science & Business Media (2005)

[10] Estrada, S.A.A.: Estimación de los Parámetros Aerodinámicos De una Mini Aeronave de Ala Fija. Master's thesis, Departamento de Mecatrónica, CINVESTAV-IPN, México, D.F., MEXICO (Agosto 2010)

[11] Fischer, M., Kennedy, D., Featherson, C.: Multilevel optimization of a composite aircraft wing using viconopt mlo. In: 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization. Atlanta, Georgia (Sep 4–6 2002)

[12] Insua, D.R.: Sobre soluciones optimas en problemas de optimización multiobjetivo. Trabajos de Investigación Operativa **2**(1), 49–67 (1987)

[13] Jameson, A.: Computational aerodinamics for aircraft design. Science **245**, 361–371 (28 July 1989)

[14] Jameson, A.: A perspective on computational algorithms for aerodynamic analysis and design. Progress in Aerospace Sciences **37**(2), 197–243 (2001)

[15] Jameson, A.: Aerodynamic shape optimization: Exploring the limits of design. In: Proceedings of KSAS 2003. Gyeongju Korea (November 14-15 2003)

[16] Junfeng, C., Wuzhou, L., Yue, W., Lili, W., Chao, W., Fei, X.: Flying qualities specifications framework for unmanned aircraft systems based on mission task elements. In: 2020 3rd International Conference on Unmanned Systems (ICUS). pp. 863–867 (2020). https://doi.org/10.1109/ICUS50048.2020.9274833

[17] Kroo, I.: Multidisciplinary Optimization Applications in Preliminary Design – Status and Directions. In: 38th, and AIAA/ASME/AHS Adaptive Structures Forum. Kissimmee, FL (Apr 7–10 1997)

[18] Kroo, I.: Innovations in Aeronautics. In: 42nd AIAA Aerospace Sciences Meeting. Reno, NV (January 5-8 2004)

[19] Li, C., Xiao, Y., lin Xu, Y., xin Peng, Y., Hu, G., Zhu, S.: Optimization of blade pitch in h-rotor vertical axis wind turbines through computational fluid dynamics simulations. Applied Energy **212**, 1107–1125 (2018). https://doi.org/https://doi.org/10.1016/j.apenergy.2017.12.035, https://www.sciencedirect.com/science/article/pii/S0306261917317531

[20] Mader, C.A., Kenway, G.K.W., Yildirim, A., Martins, J.R.R.A.: Adflow: An open-source computational fluid dynamics solver for aerodynamic and multidisciplinary optimization. Journal of Aerospace Information Systems **17**(9), 508–527 (2020). https://doi.org/10.2514/1.I010796, https://doi.org/10.2514/1.I010796

[21] Miettinen, K.M.: Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Boston, Massachusetts (1999)

[22] Naveen, R., Abinaya, B., Abinaya, D., Kalayarasi, K.: Estimation of aerodynamic coefficient for a variable span wing and a c-wing for application in personal air vehicles (pav). In: Singh, S., Raj, P., Tambe, S. (eds.) Proceedings of the International Conference on Modern Research in Aerospace Engineering. pp. 171–180. Springer Singapore, Singapore (2018)

[23] Ning, A., Kroo, I.: Multidisciplinary considerations in the design of wings and wing tip devices. Journal of Aircraft **47**(2), 534–543 (2010)

[24] Prosser, C.F., Wiler, C.D.: Rpv flying qualities design criteria. Tech. rep., DTIC Document (1976)

[25] Roskam, J.: Airplane flight dynamics and automatic flight controls. DARcorporation (1995)

[26] Skinner, S., Zare-Behtash, H.: Study of a c-wing configuration for passive drag and load alleviation. Journal of Fluids and Structures **78**, 175–196 (2018). https://doi.org/https://doi.org/10.1016/j.jfluidstructs.2017.12.018, https://www.sciencedirect.com/science/article/pii/S0889974617305054

[27] Slongo, A.G., Moraes, D.D., Mantovani, L.Q., Venturini, M.S.: Handling qualities analysis of an unmanned aircraft vehicle for agricultural spraying. Journal of Aerospace Technology and Management **12** (2020)

[28] Stevens, B.L., Lewis, F.L.: Aircraft control and simulation. Wiley-Interscience (2003)

[29] Taylor, R., Weisshaar, T., Sarukhanov, V.: Structural design process improvement using evolutionary finite element models. Journal of Aircraft **43**(1), 172–181 (January–February 2006)

[30] Toscano-Pulido, G., Coello Coello, C.A., Santana-Quintero, L.V.: EMOPSO: A Multi-Objective Particle Swarm Optimizer with Emphasis on Efficiency. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) Evolutionary Multi-Criterion Optimization, 4th International Conference, EMO 2007. pp. 272–285. Springer. Lecture Notes in Computer Science Vol. 4403, Matshushima, Japan (March 2007)

[31] Williams, W., Harris, M., et al.: The challenges of flight-testing unmanned air vehicles. Ph.D. thesis, Systems Engineering Society of Australia (2002)

[32] Xue, F., Sanderson, A.C., Graves, R.J.: Multi-objective differential evolution - algorithm, convergence analysis, and applications. In: 2005 IEEE Congress on Evolutionary Computation (CEC'2005). vol. 1, pp. 743–750. IEEE Service Center, Edinburgh, Scotland (September 2005)

[33] Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Tech. Rep. 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland (May 2001)

http://www.imavs.org/

Micro-drone autopilot architecture for efficient static scheduling

Gautier Hattenberger*[,1], Fabien Bonneval[1], Matheus Ladeira[2], Emmanuel Grolleau[2], and Yassine Ouhammou[2]

[1]École Nationale de l'Aviation Civile, Université de Toulouse, Toulouse, France
[2]LIAS, ISAE-ENSMA, Université de Poitiers, Poitiers, France

http://www.imavs.org/

ABSTRACT

This paper presents the internal architecture of a Modifiable Off-the-Shelf open-source autopilot. We show starting from a set of functional and hardware requirements why most autopilots use as a core thread a main loop acting as a non-preemptive static scheduler, reacting to external events, some solicited, some unsolicited (but expected). We explain how the type of bus used to communicate with the sensor impacts the nature of the events received from the sensors (solicited or not). We show that depending on the workload that a main loop iteration has to handle, the execution time of an iteration can be larger than the period, creating potential delays in the attitude correction. Finally, we explore the degrees of freedom that can be used to reduce the impact of these overloads by smoothing the periodic workload.

1 INTRODUCTION

There is a growing interest in open and flexible architecture for UAV system, both for research and industry, aiming at providing standards for reference implementation [1, 2, 3]. This tendency concerns single drone operation, with various level of autonomy, but also swarm and fleet control [4], sometimes based on cloud technology [5]. At a very high level, a UAV (or RPA, Remotely Piloted Aircraft) system is composed of the actual flying object(s) (*airborne segment*), a remote pilot station (*ground segment*) and a command and control link (*communication segment*) as stated by ICAO Concept of Operation [6]. Most of the described architectures, such as in [7], are considering the flight controller as a black-box, with the flight stabilization and basic navigation services provided by the manufacturer or by one of the open-source projects existing on the market, such as *PX4* [8], *ArduPilot* [9] or *Paparazzi* [10, 11]. These autopilots are often seen as modifiable off-the-shelf (MOTS) components by SME manufacturing custom drones, who have to extend these MOTS autopilot to comply with their specific needs. The internal low level timing and execution sequence, while subject to real-time constraints, and interacting with external sensors, actuators and payload control, is rarely presented or studied in the specific case of small UAV autopilots. In order to understand the internal behaviour, one has to rely on the developer documentation or even the source code. It can be a problem when trying to customize MOTS autopilots, especially when the central stabilization functions are to be addressed.

This article aims at presenting the internal software architecture of the low level functions of an open-source autopilot flight stack. We show in this paper how and why Paparazzi's central architecture is based on a central monolithic thread (*main loop*) acting as a static non-preemptive scheduler.After introducing the functions scheduling problem in section 2 and the airborne architecture used as a reference in section 3, the scheduling problem is detailed in section 4 and finally functions durations are evaluated in section 5.

2 AUTOPILOT SCHEDULING PROBLEM

The design of an autopilot consists of executing several interacting functions, each one responsible for a specific part of the complex control system that links the drone's sensors to its actuators. The connections between the functions form a Directed Acyclic Graph (DAG), which would ideally be executed in continuous time, with no delay between input and output.

However, the functions have to be executed on a hardware platform – microcontroller or microprocessor – that executes each thread sequentially, one instruction at a time, hence one function at a time for each thread. The ideally continuous control must therefore be executed in discrete time. Like for any control and command system, a loop is created by the physical process (i.e., the drone) between the actuators and the sensors, since the actuators change the state of the controlled system. Control and aerodynamics experts usually consider that in order to ensure stability, the actuators shall be controlled, given a fresh and good estimation of the state, at a rate between 400 Hz and 3 kHz for rotorcrafts such as helicopters, or multirotors, and between 50 Hz and 400 Hz for a fixed-wing aircraft. As a result, in most autopilots, the function chain starting from the state estimation and ending at setting the actuators should be executed at a rate ranging from 50 Hz to 3 kHz. For a typical rate of 1 kHz, on an embedded platform, this functional chain executed at a period of about 1 millisecond, and including computationally intensive work,

*Corresponding author: gautier.hattenberger@enac.fr

represents a non-negligible load and appears as the core part of any autopilot.

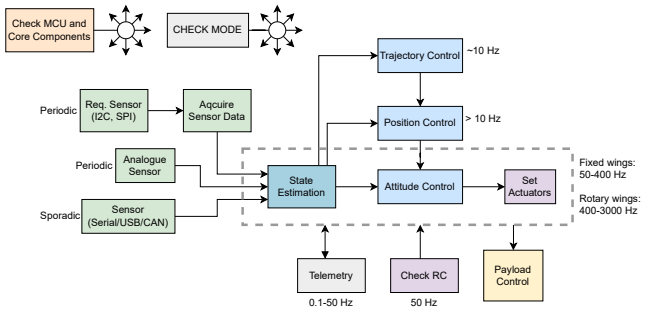


Figure 1: Representation of functions in an autopilot. The two functions in the top left corner are broadcasting to all other functions.

Figure 1 shows an extract of functions which are part of any autopilot nowadays. We can see in a common frame the chain from state estimation to actuation, which will dictate its rhythm to the control. For closely related autopilots, it can be noticed that depending on the hardware interface used to communicate, three types of constraint can appear. (1) If the sensor is analogue, then it is usually read when necessary, at a time chosen by the application (a sensor reading take less than 5 microseconds). Such a sensor can thus be read during a polling phase initiated by the application. (2) If the sensor is connected to a bus (serial, USB, CAN, etc.), then it is usually sending its readings to the autopilot periodically, but using its own clock that can drift compared to the clock used by the autopilot. We can therefore establish a minimal interval between two successive frames coming from a sensor, but we cannot choose when exactly the frame will be received. As a result, an incoming frame from such a sensor can only be considered sporadic. (3) If the bus used to communicate with the sensor is of type Master/Slave (e.g., I2C, SPI), then a delay has to be considered between the request from the Master to the response from the sensor. Since such delay can be long (dozens up to hundreds of microseconds), such functions can be decomposed in two parts: (i) the request from the Master, (ii) handling the response, where a sufficient amount of time has to elapse between the successive executions of the two functions.

Other functional chains expressed on Figure 1 have a smaller rate than the core chain state estimation to actuation. For example, the desired rate for position and trajectory control is in the order of magnitude of 10 Hz. Inputs received through legacy Radio Command (RC) is typically 40 Hz, while Datalink received from, and Telemetry transmitted to the ground station have a frequency typically ranging from 0.1 Hz to 50 Hz. The custom payload may require any frequency depending on its nature: from even higher than the core autopilot frequency, down to less than one hertz.

Note that, on each functional chain, the current state (Fly-by-Wire, Stop, Take off, etc.) shall be the same, otherwise the behaviour of the system may be inconsistent. As a result, a mode change shall occur either between two executions of the chain: at the beginning or at the end of the execution of the DAG.

Finally, MOTS autopilots usually target several types of hardware platforms, from small low-cost microcontrollers to modern heterogeneous multiprocessor on a chip boards, some barely able to host an Operating System (OS), others able to host a full POSIX PSE 54 Linux OS. As a result, the choice for Paparazzi was to be executable bare metal as well as on an OS. The same choice was made for Ardupilot, while PX4 can only be executed on a POSIX compliant OS. Regarding these functional and hardware constraints, the next section shows the internal software architecture of Paparazzi.

3 AIRBORNE ARCHITECTURE FOR THE PAPARAZZI SYSTEM

3.1 Components organization

The general architecture for any UAV system aims at providing a closed loop system with perception, decision and action. If each implementation will differ, a general good practice is to design a loosely coupled architecture, usually based on modules exchanging data through dedicated components, for example middlewares or blackboards. Such organization is found in most robotics platforms, as with the ROS/ROS2 framework [12], or in UAV specific systems like PX4 or ArduPilot. In this case study, based on the Paparazzi system, the architecture presented on Figure 2 follows the same principles.

In particular, the sensors data collected by the modules at the top right are sent to the state estimation filters (INS / AHRS blocks) through a software bus. This bus is using the publish / subscribe scheme, where data is pushed when available by the producers to the subscribing consumers. Only the common definition of messages is required to connect the elements. Note that other elements can use the same bus to create interactions between payload components for instance.

The result of the state estimation is pushed by INS/AHRS filters to a blackboard type structure, referred as *state interface* on Figure 2. The main characteristic of this interface is that new data can be pushed in any supported format (e.g. Euler angles, rotation matrix or quaternion in the case of attitude representation), while they can be retrieved from any other components within AP process and in any format as well. If a format transformation is required (e.g. from quaternion to Euler), the conversion is performed on the fly, only once, until a new update is available for the state. The available data in this interface represents position, velocity, acceleration, orientation (attitude), rotation speed, and air and wind speed. The mathematical library involved in these conversions and other algorithms have been formally verified against runtime errors [13].

http://www.imavs.org/

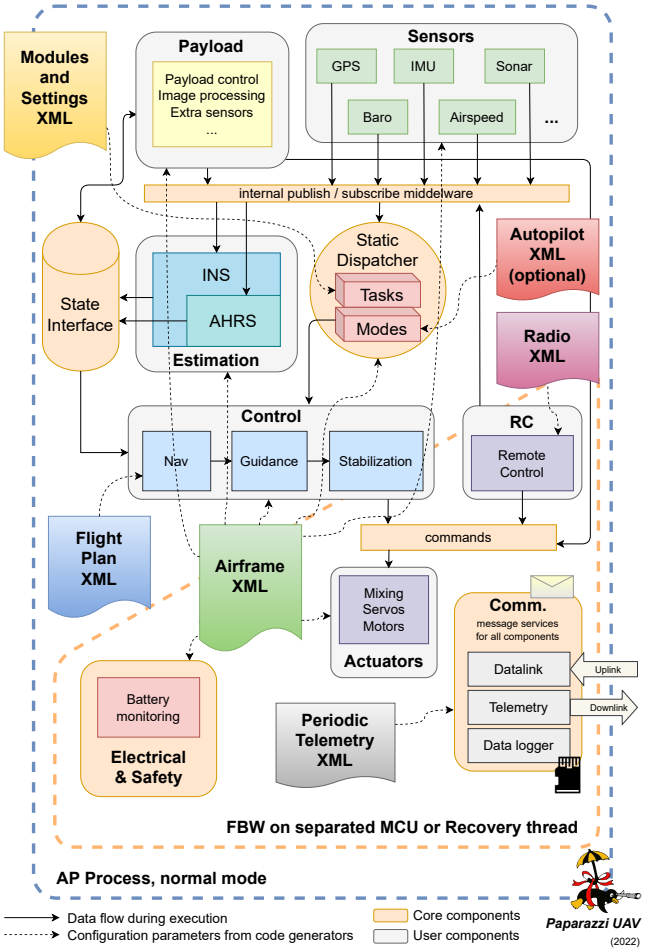


Figure 2: Airborne architecture of the *Paparazzi* system

Some other core components and services are available within the system. A *commands* array stores the normalized control vector computed by the navigation, guidance and control loops. They are ultimately applied to the physical actuators with a mixing defined in an airframe configuration file. The communication services allow sending and receiving messages from the ground, from another aircraft or from an embedded companion computer. Data logging on SD card is also possible on supported hardware. Electrical and safety monitoring are checking the energy source to apply appropriate actions in case of voltage drop.

Finally, the dispatcher is in charge of calling all the components with the correct sequence. The scheduling is statically defined during compilation based on the characteristics of the different modules. The section 4 is presenting the scheduling strategy in details.

3.2 *Configuration of the system*

The system presented Figure 2 is configured with a set of XML files, describing the airframe, the flight plan, the messages used for telemetry, the remote control and the settings (list of variables accessible from the ground control station). In particular, the airframe and the flight plan provide a list of *modules*. These modules correspond to XML files containing the following information:

- a name and a task group (note that in the document, functions can be referred to as tasks, not to be confused with threads),
- a documentation,
- a list dependencies with required or conflicting modules or functionalities, as well as functionalities provided by the module itself,
- a list of settings,
- a list of header files with public elements (functions and variables),
- a list of initialization functions,
- a list of periodic functions with the associated frequency,
- a list of event function (polling functions to check and process sporadic events),
- a list of callbacks to trigger on incoming datalink messages,
- compilation instructions with file names, compilation flags, includes, filtering options (e.g. restrict to certain firmware), etc.

The build process consists in parsing all the configuration files, solving the dependencies to find the list of required modules and generate a code corresponding to the different components, flight plan state machine and makefile to build the final binary that is finally flashed to the flight controller board.

4 Static Scheduling Approach

The scheduling approach is to divide the problem into three complementary steps:

1. For each module, resolve the dependency graph and provide an ordered list of components to load.
2. Generate initialization, periodic or event function calls, based on the task group associated to each module. Call the task groups in a predefined order to respect the closed-loop control scheme and some temporal constraints.
3. Each periodic task group consists in a static scheduler, which offset is chosen to limit the number of periodic functions being called at each iteration.

The following subsections provide details for each step.

http://www.imavs.org/

```
<dep>
 <depends>
   module1,module2|module3,@func1
 </depends>
 <provides>func2</provides>
 <conflicts>module4,@func3</conflicts>
</dep>
```

Figure 3: Example of module's dependency, functionalities are prefixed with `@` in `depends` and `conflicts` nodes, `|` corresponds to a logical *or* operator

4.1 Modules dependencies resolution

As mentioned in section 3.2, every component to be integrated to the airborne code is described in a XML module file. This file describes, for each component, its interface as the list of provided functions (see Figure 3), dependencies and conflicts as the list of required and conflicting modules or functions.

An ordered list of modules is obtained with the two-step topological sort described by Algorithm 1, and based on the depth-first search from [14]. During the first step, each time an explicit module name is specified in the `depends` node, this module is included in the recursive tree search presented in Algorithm 2. However, when it is a functionality or a Boolean expression (with `|`), it is stored for a later validation. The list of provided functionalities and potential conflicts is also stored. At the end of the first sorting algorithm, a list of modules is available, either selected by the user or required by dependency. Then the list of required functionalities is checked against the list of provided functionalities, same for logical expression and conflicts. If this check is successful, the list of modules is complete, but not properly ordered: during the first pass, functionalities are not yet instantiated with a real module. This is why a simpler second pass of the topological sorting algorithm is necessary. This time, no new modules are added and there is no need to check for functionalities and conflicts. The final result of the second pass is a fully ordered list of modules. Note that this order is not unique, and the first valid solution is kept, but it is guaranteed that a given module will always be placed after the modules it is depending on.

4.2 Temporal execution sequence

The second aspect for the correct scheduling of function calls is to guarantee a correct temporal sequence. The approach in Paparazzi autopilot is to split the functions into a predefined list of task groups, which is shown on Figure 4, in the *Init* box.

The selection and ordering of these tasks is the result of an expert-based functional analysis. Considering the *Init* phase, the initialization of the microcontroller unit (MCU) and its peripherals should be executed prior to all other tasks. Then core components of the autopilot system, and finally other

Algorithm 1 Topological sort

Require: airframe and flight plan XML files
procedure SORT MODULES
Extract list of modules selected by user from input files
Add extracted modules to a $root$ module
RESOLVE($root$)
if at least a module from resolved in conflict list **then**
fail
end if
if at least a functionality from required in conflict list **then**
fail
end if
if not all required modules and functionalities are in provided list **then**
fail
end if
Substitute functionalities or logical expression with proper modules from resolved list
RESOLVE($root$ (updated)) ▷ second pass
Return resolved list of fully ordered modules
end procedure

Algorithm 2 Dependency resolution

resolved list ← []
unresolved list ← []
conflict list ← []
provided list ← []
function RESOLVE(m) ▷ provide module as input
if m is valid for firmware and target **then**
add m to unresolved
for each dependency d in m **do**
if d is a module **then**
if d is not in resolved **then**
if d is in unresolved **then**
fail, cyclic dependency detected
else
RESOLVE(d)
end if
end if
else ▷ functionality or logical expression
add d to required list
end if
end for
add conflict to list
add provided to list
add m to resolved list
remove m from unresolved list
end if
end function

http://www.imavs.org/

components such as inputs/outputs (sensors, radio control, actuators, datalink), estimation and control, and finally the payload. Within each group, the order of execution respects the order resulting from Algorithm 1.

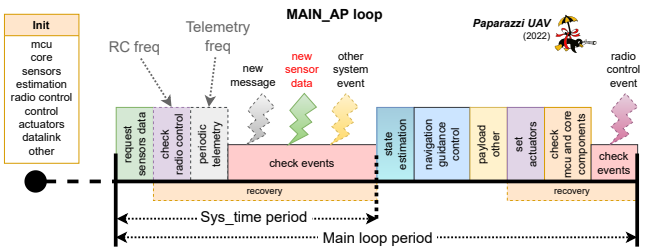


Figure 4: Main loop temporal sequence

After the initialization phase, an infinite loop with a fixed desired frequency is started. This frequency corresponds to the *Main loop period* on Figure 4. Each time the main periodic function is called, the periodic functions from each task group are called in the order presented on the figure, mostly the same as during initialization phase, but with some modifications and timing constraints. An important consideration to have is that the sensors used for state estimation on modern MAVs have digital interfaces, such as I2C or SPI, especially the Inertial Measurements Units (IMU). When called, the periodic functions for these sensors initiate a transaction to get new data. Obviously, it is not necessary to run the state estimation filters and the control stack before data availability.

Several options are available to decide when to start the rest of the flight control stack. In order to keep the estimation and control at a stable frequency, independent of the transaction time jitter, the design choice in Paparazzi is to run them after half of the main frequency. In practice, it corresponds to setting the *Sys_time period* to half of the main loop period and alternate sensor reading and control on this base frequency.

To evaluate the compatibility of this choice with existing sensors, let's consider two types of IMUs used on existing boards: the *InvenSense MPU9250* connected other I2C at 400 kHz and the *InvenSense ICM-20600* connected over SPI at 1 MHz. Each sensor sends on request 14 bytes of data and a status byte to be retrieved at each transaction, with different protocols overhead. The transaction time is presented in Table 1 and is less than 0.5 ms for these two sensors. Considering the half-period timing, it means that the main loop frequency can be set up to 1kHz in all cases, even more with fast sensors, which is compatible with typical values presented in section 2.

4.3 *Static scheduling of periodic function calls*

Scheduling the autopilot functions in this scenario requires analyzing some characteristics of the functions to be scheduled. Once the dependency has been resolved and the internal order has been defined (Algorithm 1), assuming a main loop frequency of 1kHz, functions with a period of 1ms

model	# bytes (bits/byte)	baudrate	time (ms)	max AP freq
MPU9250	15+3 (9)	400k	0.4	$\approx$ 1 kHz
ICM-20600	15+1 (8)	1M	0.13	$\approx$ 3 kHz

Table 1: IMU transaction time and maximum recommended main loop frequency

are already completely scheduled.

However, in functions that have a greater period than 1ms (a lower frequency than 1kHz), there is a degree of freedom relative to the choice of which loops will host the execution of these functions. For example, neglecting any precedence constraints, two 500 Hz functions can be executed in alternating loops such that they will never be executed in the same loop. This strategy can better distribute the workload, helping to avoid the main loop taking more than its defined period to execute.

The choice of the loop where to execute each function is not trivial. This problem is equivalent to choosing an offset for periodic functions, which has been treated in the literature and identified as closely related to a NP-hard problem [15]. Some efficient heuristic methods have already been proposed to deal with the problem for independent functions [16, 17, 18, 15] but they cannot be trivially adapted to functions subject to precedence constraints.

The current implementation is a naive algorithm that shift each periodic functions by a fraction of its period, unless user-specific instructions. The Figure 5 is showing this principle on a simplified case with three tasks running at three different frequencies. Without offset, the tasks are all called in the same loop periodically (every 6 time units in this case). With the offsets, the result is a smooth execution but without guarantee nor optimization. Also, execution time is not considered.

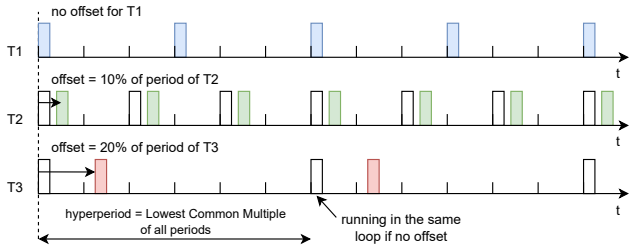


Figure 5: Task scheduling example of 3 tasks with an hyperperiod of 6 units. With no offset (white boxes), the 3 tasks are executed sometimes in the same loop. The offset of a task is computed as x times its period where $x \in [0, 1[$ is incremented by 0.1 at each new task.

http://www.imavs.org/

http://www.imavs.org/

5 PERFORMANCE EVALUATION

The performance of the autopilot has been measured by recording the execution time of the sub-functions of the autopilot during a flight. The conditions and experimental setup are as follow. The flight has been performed with a standard quadrotor frame, running the "rotorcraft" firmware of Paparazzi at 500 Hz for the main loop frequency, on the *Tawaki* board equipped with a STM32F777 at 216 MHz. In order to capture the time required by different functions, there are three main phases during the flight, that have been explicitly labeled with background colors on Figure 8:

1. The aircraft is in *KILL* mode, i.e., with no control activity. After 100 seconds, position data from a motion capture (indoor flight) are received through the datalink (red background).
2. After 120 seconds, the aircraft mode switches to *ATTITUDE* control with stabilization control activated (orange background).
3. After 210 seconds, it switches to *NAV* mode and performs a fully autonomous flight from flight plan navigation (green background).

The Figures 6, 7, 8 and 9 are all showing the average execution time (thick blue line), as well as the min and max execution time, where each data point corresponds to 1000 calls of the corresponding group. Data is stored on an on-board SD card. We measured and stored data related to 1000 calls in order to decrease the interference of the measuring time compared to the execution of the observed system. The Table 2 is summarizing the call frequency over the complete flight.

task	freq (Hz)	std
event	10007.575	0.047
sensors	500.496	0.154
radio	58.882	0.436
gnc	500.489	0.155
core	500.491	0.154
telemetry	500.483	0.154

Table 2: Execution frequency and standard deviation (std) of the periodic tasks and event functions

The polling of the event functions is running at 10 kHz, which is the resolution of the Real-Time scheduler used at low level. It can be seen on Figure 6 that the functions always takes less than 100 us, which means that it doesn't exceed the allocated time for event polling. The extra activity due to datalink incoming messages is visible at 100 seconds.

The sensor task (Figure 7) should not take a lot of time since it is mostly only requesting new data from digital sensors. The average duty time is very low, however accessing the I2C or SPI peripherals may take time in comparison depending on the underlying RTOS activity, where each peripherals' driver is running in a dedicated thread.

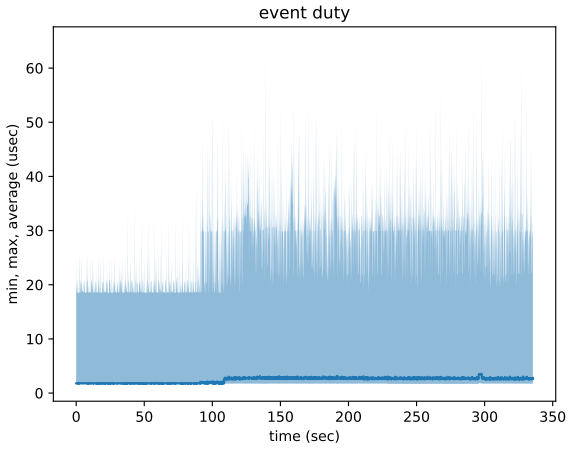


Figure 6: Duty time (average, min and max of 1000 calls in us) of the event functions during test flight. The 100 us threshold is never exceeded, which means that next event calls are not delayed. Activity is increasing with incoming data from datalink at 100 seconds.

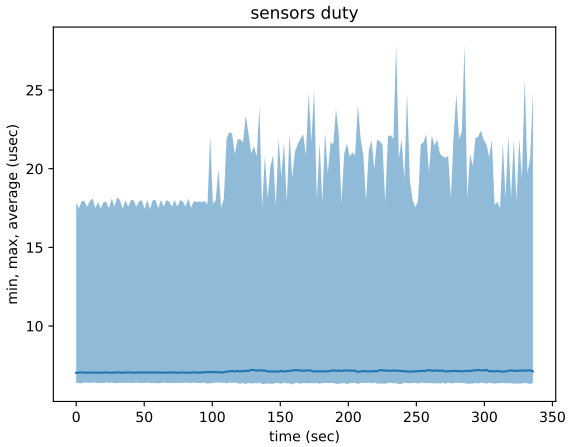


Figure 7: Duty time (average, min and max of 1000 calls in us) of the sensors periodic functions during test flight. Requesting new data takes less than 25 us.

The activity of the GNC (grouping estimation, navigation, control and payload tasks) is the most interesting, as Figure 8 shows that the execution time is varying a lot according to the flight phase. This setup is using an advanced controller that computes dynamically a control allocation with an optimization algorithm for the attitude control [19]. Therefore, the control activity is rising from 20 us to almost 80 us when the aircraft is flying from radio control (orange part), and even more when guidance loop is activated for navigation (green part). The maximum execution time still remains low (less than 120 us) compared to the main loop period of 2000 us (500 Hz).

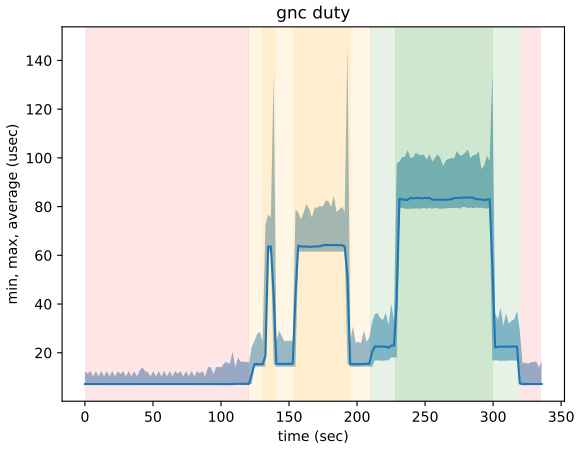


Figure 8: Duty time (average, min and max of 1000 calls in us) of the estimation, navigation, control and payload functions (GNC) during test flight. Background colors indicates the flight mode: red for kill mode (no activity), orange for direct attitude control (stabilization loop), green for flight plan navigation (full guidance and stabilization). Darker color indicates that the quadrotor is flying, with higher CPU duty.

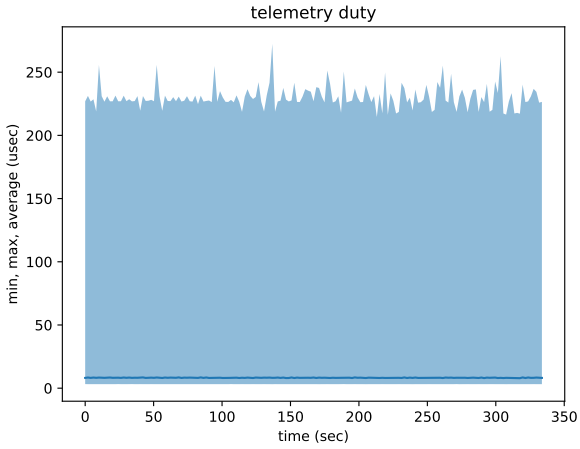


Figure 9: Duty time (average, min and max of 1000 calls in us) of the event functions during test flight. The average duty stays very low ($<$ 10 us) compare to the maximum values (around 225 us), this is due to the sporadic nature of message sending in each interval. When a message is sent, most of the duty time is spend in waiting access to the UART buffer, shared with the peripheral thread.

Finally, the last set of functions presented here is the telemetry group (see Figure 9). For similar reasons than with the sensors functions, the average duty time is very low ($<$ 10 us). These functions are called at 500 Hz, but the messages themselves are scheduled at much lower frequency, typically between 0.1 and 50 Hz (see Figure 1). As mentioned before, each MCU peripheral, including the UART connected to the modem, is running in a dedicated thread, and accessing the shared memory buffer, protected by a mutex, involves calls to the scheduling functions of the underlying RTOS. As a result, the maximum execution time of this group is the longest, going up to 270 us, although no processing related to navigation is involved. The other functions of the autopilot that are not presented on these figures are showing similar trend than the sensors or telemetry task and are not much interesting to be discussed.

The overall MCU load, including the autopilot thread and all other peripheral threads are measured around 10 % by the RTOS.

6 Conclusion

Nowadays autopilots are MOTS, meaning that personalized modules can be added to an existing autopilot. Moreover, experimentation shows that, while an autopilot control loop running at 500 Hz should mainly support the basic functions of the autopilot, there are two situations where some interference between execution of functions may postpone the execution of some core functions. First, if a specific module given by the user of the autopilot as a MOTS is computationally intensive, the additional functions may postpone some functions of the autopilot. Moreover, as told in introduction, some controller may be executed at a higher rate. In this case, the execution of some loops may postpone the execution of the next loop, creating delays and jitters, which usually have a negative impact on quality of control [20].

We have identified, as a way to avoid this type of interference, a means which consists in adjusting offsets of periodic functions with a period higher than the main loop frequency. Nevertheless, the current naive implementation doesn't other guarantees and some academic work has to be done for offset assignment methods to take all constraints into account, especially precedence constraints with a minimal inter-release interval.

Finally, the measurement framework is in place, and can be adapted to all the systems that drone manufacturers would like to base on Paparazzi. As an autopilot, Paparazzi is highly configurable.The perspective is to provide a framework allowing drone manufacturers to (1) measure the performances of the functions during flight, and (2) generate a static scheduler able to balance the load by adjusting offsets, while showing that at least in average, the main controller loop is able to be fully executed within its allocated window. For the purpose of being adaptable to every possible use, we are developing a bridge between files generated by the compilation of Paparazzi, and a Domain Specific Language (DSL), based on an extension of AADL proposed in the COMP4DRONES European project. This DSL will allow custom autopilots based on Paparazzi to be scheduled, and analyzed by the framework discussed in this paper.

http://www.imavs.org/

http://www.imavs.org/

ACKNOWLEDGEMENTS

This work was partially funded by ECSEL JU program as a part of the Comp4Drones project with grant number "826610".

REFERENCES

[1] Jose Luis Sanchez-Lopez, Ramón A. Suárez Fernández, Hriday Bavle, Carlos Sampedro, Martin Molina, Jesus Pestana, and Pascual Campoy. Aerostack: An architecture and open-source software framework for aerial robotics. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 332–341, 2016.

[2] Matheus Ladeira, Yassine Ouhammou, and Emmanuel Grolleau. Towards a modular and customisable model-based architecture for autonomous drones. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1127–1128, 2020.

[3] Mahmoud Hussein and Reda Nouacer. Towards an architecture for customizable drones. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 67–72, 2020.

[4] Juan A. Besada, Ana M. Bernardos, Luca Bergesio, Diego Vaquero, Iván Campaña, and José R. Casar. Drones-as-a-service: A management architecture to provide mission planning, resource brokerage and operation support for fleets of drones. In *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 931–936, 2019.

[5] Chen Hong and Dianxi Shi. A control system architecture with cloud platform for multi-uav surveillance. In *2018 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 1095–1097, 2018.

[6] ICAO RPAS Panel. ICAO RPAS Concept of Operations, 2021.

[7] Georgios Kakamoukas, Panagiotis Sarigiannidis, and Ioannis Moscholios. High level drone application enabler: An open source architecture. In *2020 12th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, pages 1–4, 2020.

[8] PX4. https://docs.px4.io/master/. Accessed: 2022-04-04.

[9] ArduPilot. https://ardupilot.org/. Accessed: 2022-04-04.

[10] Pascal Brisset, Antoine Drouin, Michel Gorraz, Pierre-Selim Huard, and Jeremy Tyler. The Paparazzi Solution. In *MAV 2006, 2nd US-European Competition and Workshop on Micro Air Vehicles*, page pp xxxx, Sandestin, United States, October 2006.

[11] 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.

[12] Ros/ros2. https://docs.ros.org/. Accessed: 2022-04-04.

[13] Baptiste Pollien, Christophe Garion, Gautier Hattenberger, Pierre Roux, and Xavier Thirioux. Verifying the Mathematical Library of an UAV Autopilot with Frama-C. In *26th International Conference on Formal Methods for Industrial Critical Systems - FMICS 2021*, Paris, France, August 2021.

[14] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001.

[15] Matheus Ladeira, Emmanuel Grolleau, Fabien Bonneval, Gautier Hattenberger, Yassine Ouhammou, and Yuri Hérouard. Scheduling offset-free systems under fifo priority protocol. In *34th Euromicro Conference on Real-Time Systems (ECRTS)*. Schloss Dagstuhl, 2022.

[16] Joël Goossens. Scheduling of offset free systems. *Real-Time Systems*, 24(2):239–258, 2003.

[17] Mathieu Grenier, Joël Goossens, and Nicolas Navet. Near-optimal fixed priority preemptive scheduling of offset free systems. In *14th International Conference on Real-Time and Networks Systems (RTNS'06)*, pages 35–42, 2006.

[18] Mathieu Grenier, Lionel Havet, and Nicolas Navet. Pushing the Limits of CAN - Scheduling Frames with Offsets Provides a Major Performance Boost. *4th European Congress on Embedded Real Time Software (ERTS 2008)*, 2008.

[19] Ewoud J. J. Smeur, Guido C. H. E. de Croon, and Qiping Chu. Cascaded incremental nonlinear dynamic inversion control for MAV disturbance rejection. *CoRR*, abs/1701.07254, 2017.

[20] Zakaria Sahraoui, Emmanuel Grolleau, Driss Mehdi, Mohamed Ahmed-Nacer, and Abdenour Labed. Predictive-delay control based on real-time feedback scheduling. *Simulation Modelling Practice and Theory*, 66:16–35, 2016.

Hover Controller Design and Implementation for a Dragonfly-like Flapping Wing

A. Mohammad Lashgari, B. Abolghasem Naghash*
Amirkabir University of Technology , Tehran

http://www.imavs.org/

Abstract

The aim of this paper is modelling, simulating and designing a controller for a micro aerial vehicle (MAV) as well as implementing an innovative scheme for it. The MAV scheme inspired by the dragonfly using clap and fling mechanism with an active rigid abdomen. The gearboxes are responsible for moving two pairs of four flying wings in front and back of the MAV. Also, change in the motors speed generates differential thrust to create a control pitch moment. The linearization about a hover point is performed to analyze the motion and design an LQR controller. Moreover, validity of the linearized equations is verified by comparing the responses of linear and nonlinear models. The theoretical results are experimented with a hardware-in-the-loop testbed. The experimental results demonstrate good agreement between theoretical and validated responses along with accurate and robust hovering at the desired point.
Keywords: Flapping wing MAV, Active rigid abdomen, Hover controller design, LQR controller.

1 Introduction

Wonderful flights of insects and birds have been noticed by many researchers. Each category of birds and insects has special mechanism for guiding, controlling and producing lift and thrust. Among all flight maneuvers of insects and birds hovering in stationary point [1] with high stability is the determining factor for quality of flight. Hover is the unstable mode of flight and for a flapping wing MAV's is very important [2, 3, 4, 5]. The main concern in this paper is to control the pitch angle for maneuvering in hover for a dragonfly-like, MAV having clap and fling mechanism. Hereafter, we call it as our MAV for short.
There is a valuable work for modeling the nonlinear aerodynamics of a flapping wing in[6] . Precise relationships to calculate lift and drag theoretically and experimentally are introduced in [1] that are used in Section 4.

*Email address:Naghash@aut.ac.ir

To review the role of an active rigid abdomen in a flapping wing MAV there has not been much theoretical or modeling or simulation work, only in reference [7] there are some experiments for a real moth (Manduca sexta). In reference [8] for a butterfly whose abdomen creates a control moment there are some theories and control design using sliding mode method but no implementation is presented. In reference [5] there are valuable theoretical work and linear controller design for the hover of a flapping wing MAV without the abdomen but unfortunately there is no implementation. DelFlyII MAV has one rotational degree of freedom to produce thrust [6, 9]. The advantage of DelFlyII in producing lift is that it uses a rotation of just one motor and in our paper we used this scheme.
In general, forces and moments created by rudder and elevator are not enough to compensate for atmospheric disturbances. Performance of rudder and elevator depends on the input airflow therefore when decreasing the altitude in hover the role of rudder and elevator can be reversed[10]. This subject makes the autopilot control loops need more sensors and can generate uncontrollable flight situations. Therefore deleting rudder and elevator in a flapping wing MAV's is a good point [10]. Since the abdomen is relatively large, it slows down the dynamics and decreases maneuverability. This type of abdomen can be sensitive to aerodynamic disturbances and creates problems in flight.

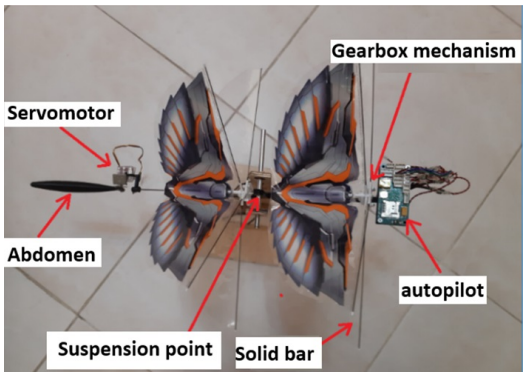


Figure 1: The proposed dragonfly-like MAV with a new structure.

After reviewing the MAV's studies, we see the modeling and simulation of a dragonfly with an active rigid ab-

domen is important and fills the gap. In this paper a new structure consisting of two pairs of mechanisms used in DelFlyII with separated motors for front and back wings like a dragonfly and a rigid abdomen mechanism which can rotate are used as shown in Figure 1.

2 Modelling the multibody dynamics

In this section using Kane's method [11, 12, 13] , the multi-body modelling for the motion of a dragonfly-like MAV will be performed. Dragonfly-like MAV has different parts, a moving rigid abdomen, and a thorax. The kinematics of thorax and abdomen(velocities, accelerations, and angular velocities) will be derived in the inertial frame. Then after calculating external and inertial forces, equations of motion for longitudinal mode will be derived.

2.1 Introducing the structure for the dragonfly-like MAV

In this paper, we used the DelFlyII idea, flapping-wing which is of the type of clap and fling mechanism according to figure 2 is utilized. The recommended structure consists of two pairs of flapping-wing MAV which has eight wings as shown in figure 2. Each flapping-wing separately acts using the change in motor rpm to create lift. Thus, the deferential lift between front and back generates a pitching moment. Each wing from the root of the wing and the leading edge is fixed to fiber carbon bars and the trailing edge of each wing is free with no support. In some plans two more rigid bars are used in each wing. When wings 1 and 2 get close to each other the rigid bars in the leading edges would reach together fast and the trailing edges would reach together slowly. In this situation, the front of wings 1 and 2 are closed and the air is trapped within two wings and it has no way other than leaving from the back and in this way a thrust is produced. This complex motion of wings is done using a motor and a four-link mechanism for wings 1 and 2 and separately for wings 3 and 4.
Another difference between DelFlyII FW-MAV and our FW-MAV is that DelFly has rudder and elevator and it utilizes the flapping mechanism just to produce thrust. As we see from figure 1, the active rigid abdomen with one degree of freedom has just a dynamic role through control moment and despite DelFlyII MAV does not have any rudder nor elevator. Also, the gearbox for this MAV is shown in figure 3.

We have shown wing gearbox mechanism in figure 3. The bars OA, AB, and CB with fixed supports of O and C make a four-link mechanism. In section 3, we will analyze and simulate the gearbox.

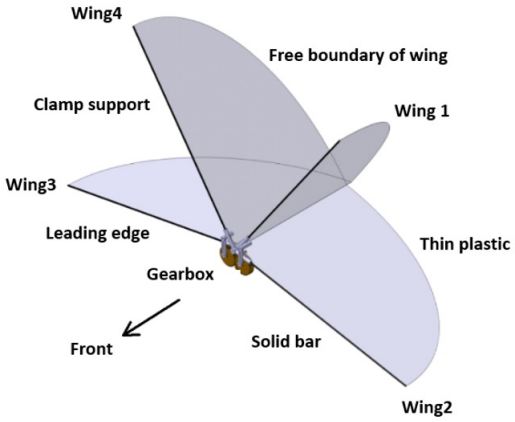


Figure 2: Clap and fling section for DelFlyII MAV.

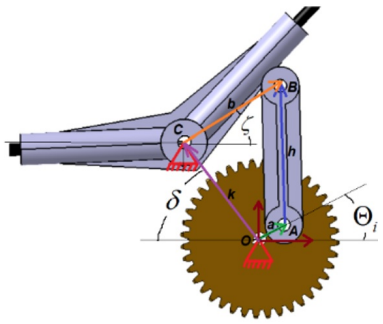


Figure 3: The gearbox and the four-link mechanism.

2.2 Free body diagram kinematics for dragonfly-like MAV

In figure 4, the free body diagram for dragonfly-like MAV is shown. In figure 4, the unit vectors $(\hat{n}_1, \hat{n}_2, \hat{n}_3)$ of inertial frame, $(\hat{b}_1, \hat{b}_2, \hat{b}_3)$ of the frame to fixed dragonfly thorax, and $(\hat{t}_1, \hat{t}_2, \hat{t}_3)$ of the frame fixed to abdomen are defined. There is a one degree of freedom hinge between the abdomen and the thorax. The distance between center of gravity of thorax and H hinge l_t and the distance between hinge is H and center of gravity of abdomen is l_a.
To derive the equations, we ignore the head mass, wing mass, and wing moment of inertia of the MAV. The dragonfly MAV in longitudinal mode has four degrees of freedom, two degrees of freedom for translation, one for rotation of thorax and one for abdomen angle Θ_a. In reference [13] , we have detailed the derivation of equations of motion.
In figure 5, the drag force is denoted by D Also the lift force, velocity vector, angle of attack, pitch angle are indicated by L, V, and α, Θ respectively.

http://www.imavs.org/

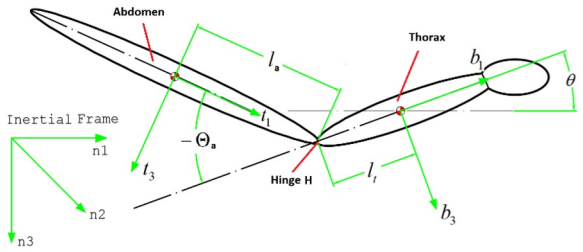


Figure 4: The thorax, abdomen, and inertial frames for dragonfly-like MAV.

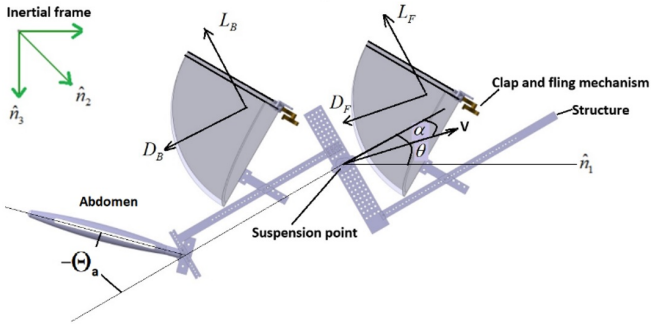


Figure 5: The dragonfly-like MAV at hover (hardware in the loop).

2.3 Longitudinal mode equations using Kane's method

According to Kane's method by after some lengthy algebraic manipulation, the longitudinal mode equations of motion can be derived as the following[19]:

$$
\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} \dot{U} \\ \dot{W} \\ \dot{Q} \end{bmatrix} + \\
+ \begin{bmatrix} -(m_a+m_t)QW - m_a l_t Q^2 ... \\ -m_a l_a \cos\Theta_a (Q+Q_a)^2 ... \\ -m_a l_a \sin\Theta_a \dot{Q}_a \\ (m_a+m_t)QU ... \\ +m_a l_a \sin\Theta_a (Q+Q_a)^2 ... \\ -m_a l_a \cos\Theta_a \dot{Q}_a \\ m_a QU(l_t + l_a\cos\Theta_a) ... \\ -m_a l_a l_t \sin\Theta_a Q_a (Q_a+2Q) ... \\ -m_a l_a \sin\Theta_a QW ... \\ +(-I_{ayy} - m_a(l_a^2 + l_a l_t \cos\Theta_a) - m_a l_a^2)\dot{Q}_a ... \\ +C_{mq}Q \end{bmatrix} \\
+ \begin{bmatrix} -(m_t+m_a)g\sin\theta \\ (m_t+m_a)g\cos\theta \\ m_a g l_t \cos\theta + m_a g l_a \cos(\theta+\Theta_a) \end{bmatrix} \\
+ \begin{bmatrix} -D_F - D_B \\ -L_F - L_B \\ L_F b_{FW} - L_B b_{BW} \end{bmatrix} = 0 \tag{1}
$$

In equation (1), values used for m_{ij} of inertial matrix are as follows:

$$
\begin{aligned}
m_{11} &= m_{22} = -(m_a + m_t) \\
m_{12} &= m_{21} = 0 \\
m_{13} &= m_{31} = -m_a l_a \sin\Theta_T \\
m_{23} &= m_{32} = -m_a(l_t + l_a\cos\Theta_a) \\
m_{33} &= -I_{ayy} - I_{tyy} - m_t(l_t^2 + 2l_a^2 + 2l_a l_t \cos\Theta_a)
\end{aligned} \tag{2}
$$

In equation(1), the parameter D_F is front wing drag force, D_B is back wing drag force, L_F is front wing lift force, L_B is back wing lift force, and Q_a is the angular velocity of the abdomen which is produced using a servo. The above lift and drag forces are shown in figure 5. b_{FW} is lift moment arm and is equal to distance between the lift force effective point of front wing and the suspension point for the MAV (cg of the whole MAV is suspended on the base). Similarly b_{BW} is the distance between the effective lift force point for back wing and the suspension point of the MAV. Therefore $\mathrm{L}_F\mathrm{b}_{FW}$ and $\mathrm{L}_B\mathrm{b}_{BW}$ are moments produced from back and front wings lift forces. The difference between these values $\mathrm{L}_F\mathrm{b}_{FW} - \mathrm{L}_B\mathrm{b}_{BW}$ is a differential thrust which creates a pitch moment. Equation (1) is a nonlinear equation for longitudinal mode for the dragonfly-like MAV. The first matrix in that equation is the inertia and the second one matrix in equation 5 consists of non-gravitational forces and the third one matrix consists of gravitational forces. The fourth matrix consists of external forces. L_F and L_B are lift forces for front and back wings, respectively. D_F and D_B are drag forces for front and back wings, respectively.

http://www.imavs.org/

http://www.imavs.org/

3 Simulating dragonfly flapping-wing gearbox

In this section first we repeat simulation for DELFLYII MAV gearbox and using the dynamic constraints, we will get a relationship for wing angle and its derivatives with respect to the motor rpm. The basis for the DELFLYII gearbox is a four-link mechanism shown in figure 3. By using kinematic relationship for links, we can derive the following relationship for the gearbox:

$$\begin{aligned}\dot{\zeta} = \dot{\Theta}_i(a/b)(&\cos\Theta_i(k\cos\delta - a\sin\Theta_i + b\sin\zeta)\\ &+\sin\Theta_i(a\cos\Theta_i + k\sin\delta - b\cos\zeta))\\ &/(\cos\zeta(k\cos\delta - a\sin\Theta_i + b\sin\zeta)\\ &+\sin\zeta(a\cos\Theta_i + k\sin\delta - b\cos\zeta))\end{aligned} \tag{3}$$

Equation (3) gives the angular velocity of the wing as a function of mechanism parameters and motor rpm. After simulating this gearbox in MATLAB/Simulink the angular velocity of wing as output with respect to time is plotted in figure 6. By using equation (3) and integrating and differentiating in Simulink, we can get the parameters ζ, $\dot{\zeta}$,$\ddot{\zeta}$. These parameters are needed to calculate the drag and lift forces in section 4.

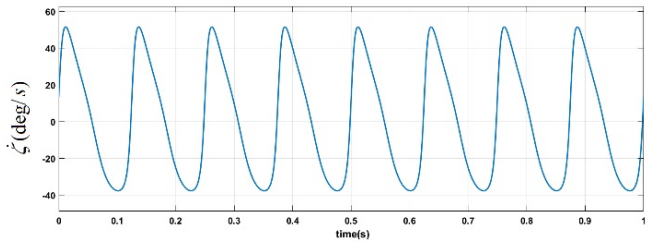


Figure 6: Simulation for flapping-wing gearbox mechanism.

4 Aerodynamic forces calculation (lift and drag)

In reference [6] for an element of the wing, lift and drag forces in body frame are calculated and then using integration in the total area of the wing, a resultant lift and drag forces at each instant are calculated in MATLAB. The resultant quasi-steady aerodynamic forces for a wing rectangular element from the leading edge up to trailing edge in reference [6] is obtained as:

$$d\vec{F} = d\vec{F}_{inertial} + d\vec{F}_{circ} + d\vec{F}_{addmass} - d\vec{F}_{visc} \tag{4}$$

The first portion, inertial forces, is a function of wing mass, fluid mass, wing velocity in z and x directions and the angular velocity of wing around the rigid support at the leading edge. We should mention that side forces in this paper are neglected, i.e. in y-direction we have no force, no acceleration, and no velocity or they do not have any effect on lift and drag. The second portion is circular forces $d\vec{F}_{circ}$, in the third portion we have general forces like $d\vec{F}_{added\,mass}$. Air flow between two wings is not uniform and its velocity changes from a maximum to zero while entering. Therefore, added air between two wings has acceleration and it exerts a force to the wings. It has been explained in reference [6]. The fourth portion is the viscous force which depends on the airflow viscosity that creates a drag force. The obtained forces in equation (4) are forces exerted on a rectangular element of the wing in body frame. To calculate the applied forces on the whole wing, we should integrate forces for this rectangular element in MATLAB software.
In this way the total lift and drag forces in body frame are calculated. After simulating all the aerodynamic relationships reference [6] in Simulink for a motor rpm of 8Hz, it gives the lift and drag for two work cycles in Figures 7 and 8, respectively.

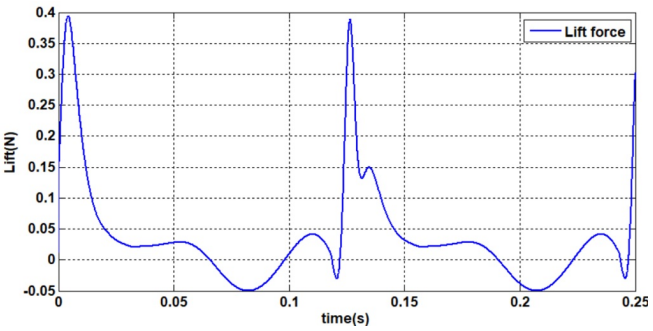


Figure 7: Lift force for rpm of the motor at 8Hz and pitch angle of 80° .

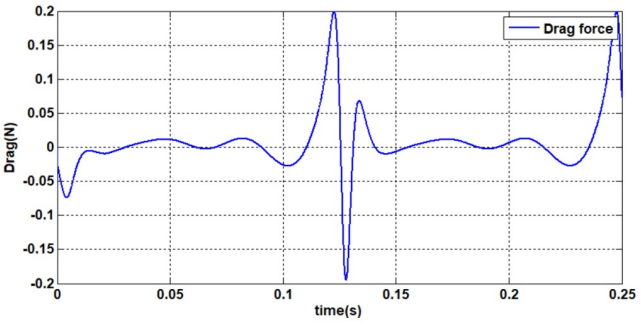


Figure 8: Drag force for the motor rpm at 8Hz and pitch angle of 80° .

As seen from the lift curve in part of a cycle the lift force is even negative but the average lift force in one cycle is positive. Similarly, for drag curve we see that in some portion of a cycle the drag force is even positive but the average drag in one cycle is negative; therefore, positive thrust is created.

http://www.imavs.org/

5 Linearization around hover point for longitudinal mode on a suspension base

In this section, we want to derive the equations for the MAV pitch angle with one rotational degree of freedom on the suspension base and then by linearizing, we will get the linear equations. Since the base is fixed, we can ignore U and W components of the velocity and put zero for U and W. Thus, adding the aerodynamic moment from C_{mq} and doing some algebraic manipulation in equation(1), we will get the following equation:

$$\begin{aligned}&(-I_{ayy}-I_{tyy}-m_a(l_t^2+l_a^2+2l_al_t\cos\Theta_a))\dot{Q}=\\&\ m_al_al_t\sin\Theta_aQ_a(Q_a+2Q)-m_agl_t\cos\theta\\&-m_agl_a\cos(\theta+\Theta_a)\\&-(-I_{ayy}-m_a(l_a^2+l_al_t\cos\Theta_a))\dot{Q}_a\\&+C_{mq}Q-b_{FW}\times L_F+b_{BW}\times L_B\end{aligned}\tag{5}$$

In equation (5), L_F is lift force for front wings, L_B is lift force for back wings,b_{FW} , b_{BW} are the distance between the effective point of lift of front (back) wings and the suspension point. The suspension point is assumed to be located on the CG of the MAV.
Assuming the state variables $x_1=\theta$ and $x_2=\dot{\theta}$, equation (5) after some algebraic manipulation can be written as follows:

$$\begin{cases}\dot{x}_1=x_2\\ \dot{x}_2=(m_al_al_t\sin\Theta_aQ_a(Q_a+2x_2)...\\ \quad -(-I_{ayy}-m_a(l_a^2+l_al_t\cos\Theta_a))\dot{Q}_a...\\ \quad -m_agl_t\cos x_1\ -m_agl_a\cos(x_1+\Theta_a)...\\ \quad -L_F\times b_{FW}+L_B\times b_{BW})+C_{mq}x_2...\\ \quad /(-I_{ayy}-I_{tyy}-m_a(l_t^2+l_a^2+2l_al_t\cos\Theta_a))\end{cases}\tag{6}$$

The aim of this portion is to linearize equation (6) around the hover point (the hover is defined as the point which the MAV can suspend itself on the desirable pitch angle which is assumed to be 80° in this paper). The main challenge to linearize is due to not having an explicit relationship between lift force and the longitudinal angles and actuator input (the system consisting of motor and the gearbox which rotates the wings). For linearization, we should differentiate the lift force with respect to θ and $\dot{\theta}$ shown in section 4. On the other hand, ζ in equation (3) and its derivatives in the gearbox simulation equation are nonlinear. These two issues make calculation of Jacobian for linearization impossible. Therefore, we cannot derive an explicit linear relationship for any desirable angle of hover. As an example, we linearize the system around a hover angle of 80°.
To resolve the linearization challenge, we used simulation for aerodynamics in Simulink twice around 80° (with ±5°). According to figure 9, we see that for ±5° deviation of hover, the average lift in each flapping cycle changes less than one percent, therefore we can ignore the lift derivative with respect to angle θ of the MAV. Thus, we can have:

$$\partial L_B/\partial x_2=\partial L_F/\partial x_2=\partial L_B/\partial x_1=\partial L_F/\partial x_1\simeq 0\tag{7}$$

Another problem in linearization is calculating the lift and drag force derivatives with respect to gearbox motor rpm. To resolve this problem, we used the theorem of average function for each flapping cycle. To do this, a program in MATLAB is written which changes the motor rpm from 1Hz to 15Hz with a step of 0.1 for lift and drag simulation. A second-degree curve was fitted to the results. Then using the interpolation, the results are shown in figure 10.
By interpolation, the average lift force function with respect to rotational frequency of the motor can be derived as follows (150 times aerodynamic simulations for different rpm of the motor were run):

$$L=0.0008f_{motor}^2-0.0025f_{motor}\tag{8}$$

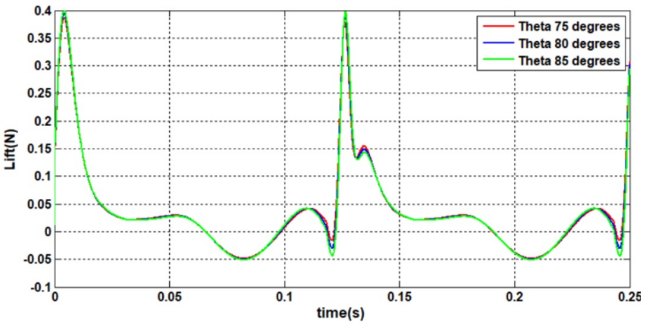


Figure 9: Change of lift force in two cycles for different pitch angles .

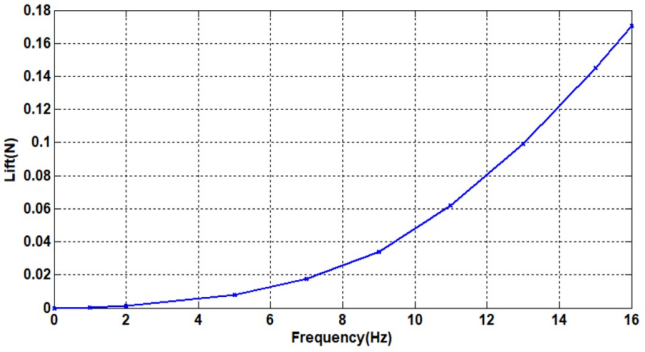


Figure 10: Lift force with respect to motor rpm of flapping-wing MAV .

Therefore, the challenge of the differentiating lift with respect to rpm of the motors for linearizion will

http://www.imavs.org/

be eliminated by using equation (8). In equation (6), by equating zero, the trim point will be obtained:

$$\begin{aligned} x_1 &= \pi/2 \\ x_2 &= 0 \end{aligned} \tag{9}$$

The physical concept for this equilibrium position for the MAV with abdomen which was a two-body system was similar to the equilibrium of a double pendulum. But if we want to hover around 80°, we should assume the equilibrium as 80° and we should transfer the state space equations to this point. The general form of the state space is as follows:

$$\dot{x} = Ax + Bu \tag{10}$$

After transferring to the trim point of 80°, we can write the state equations as follows:

$$\dot{x} = A \begin{bmatrix} x_1 - \frac{80\pi}{180} \\ x_2 \end{bmatrix} + Bu \tag{11}$$

In equation (11), $u = [\dot{Q}_a \, \Delta L \, \Theta_a]$ is the input vector which includes $\dot{Q}_a$ that is the angular acceleration applied to the abdomen of the MAV through the servomotor and creates the control moment and $\Delta L = L_B - L_F$ creates the differential lift.
We have assumed the trim state (equilibrium) is the state in which the MAV has a pitch angle of 80° and the abdomen is along the thorax ($Q_a = -\Theta_a = 0$). The numerical values of matrices A and B based on the geometrical and inertia /mass specifications and equilibrium state values in Jacobian can be obtained as follows (according to values in Table 1 for the designed MAV):

$$A = \begin{bmatrix} 0 & 1 \\ -19.88138 & 0 \end{bmatrix} \tag{12}$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ -0.126106 & -38.28667 & -5.0859346 \end{bmatrix} \tag{13}$$

According to equation (11), we see the linearized equations for the MAV with an active rigid abdomen and clap and fling mechanism for hovering around 80° on the suspension base were obtained:

$$\begin{aligned} \dot{x} = &\begin{bmatrix} 0 & 1 \\ -19.88138 & 0 \end{bmatrix} \begin{bmatrix} x_1 - \frac{80\pi}{180} \\ x_2 \end{bmatrix} \\ &+ \begin{bmatrix} 0 & 0 & 0 \\ -0.126106 & -38.28667 & -5.0859346 \end{bmatrix} \begin{bmatrix} \dot{Q}_a \\ \Delta L \\ \Theta_a \end{bmatrix} \end{aligned} \tag{14}$$

To verify the linearization, the simulation response for linear and nonlinear models are compared in figures 11 and 12. For the MAV to be stable at angle of 80°, there should be a moment which is from the difference in motor rpm which results in differential thrust at each flapping cycle. By doing some simulations, this value would be -0.0037 (the minus sign is for the direction of the moment). As a physical explanation, to stabilize the open-loop, the force from the back wings should be more than that of the front wings. It creates the differential moment and that would result in the MAV to be fixed at 80° (otherwise the MAV would go in the vertical direction). By using this value for the moment, differential lift force can be obtained from equation (15):

$$L_B - L_F = \frac{0.0037}{0.04} = 0.0925N \tag{15}$$

Assume that we want the rpm of the back wings to be 15Hz, this motor rpm based on the simulation results using equation (14) would mean that the average lift force in each flapping cycle would be equal to $L_B = 0.1425N$. Therefore, according to equation (8), the average lift force for the front wings in each flapping cycle would be equal to 0.05N. We can assumed that the MAV is on the suspension base and the summation of the lift force of front wings and back wings would be less than or equal to total weight:

$$L_F = 0.05N \tag{16}$$

This average value for wing lift force based on simulation results or equation (14) would happen at motor rpm for front wings with a frequency of 9.4Hz. These two motor rpm would create a hover of 80 degrees in the open-loop case.

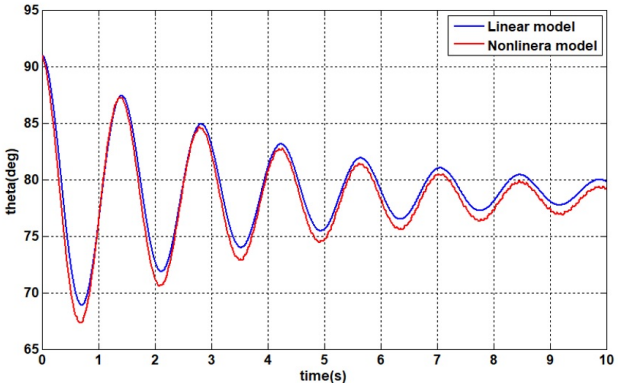


Figure 11: The comparison of body pitch angle in linear and nonlinear cases .

In summary, the control input was obtained for equilibrium conditions for the MAV at hover with an rpm frequency of front motor rpm at 9.4Hz and back motor rpm at 15Hz and an angle of attack of 25° and a pitch angle of 80°. We should mention that in simulation, we have added C_{mq} damping to the simulated equations similar

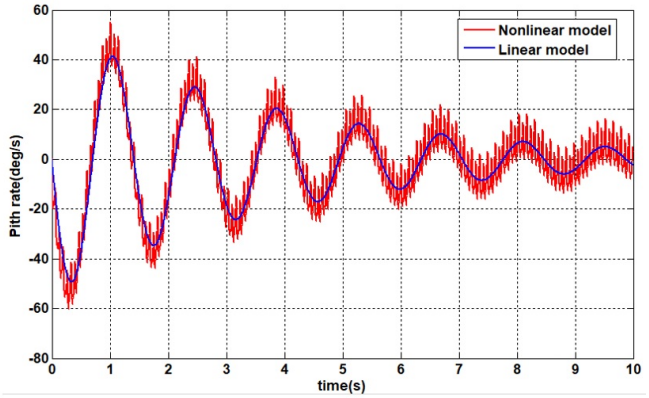


Figure 12: The pitch rate of the body in linear and nonlinear case .

to the case in real flight. The air viscosity would damp oscillation of pitch angle, the effect of C_{mq} damping can be seen in figures 11 and 12 time responses. We have assumed the initial conditions for pitch angle at $\theta_0 = 91°$ in linear and nonlinear simulations.
As we see in figure11, the pitch angle of the MAV would oscillate around 80°. In figure 12 we see the pitch rate as the output of the simulation for the MAV and we see that a frequency based on flapping would ride on nonlinear response. As we see the nonlinear and the linear responses are close to each other and in this way the linearization process and the assumptions made to resolve the challenges are verified.

6 Designing and manufacturing the dragonfly-like MAV and introducing its hardware

The values of parameters of the dragonfly like MAV and the specifications of the system are shown in Table 1.

In figure 13 the dragonfly like MAV is shown at hover position at an angle of 80°.

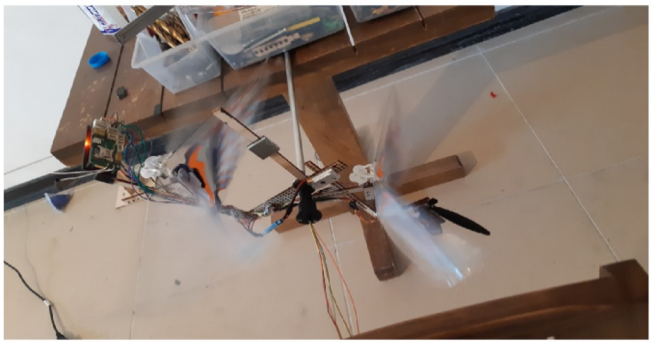


Figure 13: The dragonfly-like MAV at hover (hardware in the loop).

a	0.0045	m
b	0.00647	m
b_{BW}	0.04	m
b_{FW}	0.04	m
d	0	m
h	0.0215	m
I_{tyy}	0.000569	$kg.m^2$
I_{ayy}	1.35e-5	$kg.m^2$
l_t	0.16	m
l_a	0.055	m
M_{wing}	0.00029	kg
m_t	0.11	kg
m_a	0.01	kg
R_w	0.0137	m

Table 1: parameters of the dragonfly like MAV used in simulation.

To implement the controller, we need some elements like micro brush motor, the speed controller board for micro brush motor, Apogee autopilot board, Paparazzi software, DELFLY gearbox system and the wings, a servo motor, lithium battery and slip ring, and the USB serial converter. These hardware elements are shown in figure 14.

7 Implementing hardware in the loop controller by using Paparazzi Apogee board V1

In previous sections of this paper, the equations of dragonfly like MAV were simulated in Simulink MATLAB. In this section, the Simulink file was created such that we can relate the Simulink with Paparazzi software real time and we can send the control commands to Apogee board hardware and get sensors feedback. Implementing the control algorithm on a software consists of two main sections:
I- software and hardware of autopilot
II- Simulink software and its interface with autopilot
In this section first we explain the needed hardware to control the MAV. To control the actuators of the MAV (two micro brush motors for back wings and front wings and servomotor for moving the abdomen), which are of DC motor type, we use a hardware named ESC or electronic speed controller. This hardware consists of voltage and PWM signal as inputs and voltage to the DC motor as the output. The width of the input PWM signal to hardware can be changed from 1000ms to 2000ms which and it correspond to zero speed and the maximum speed of the motor, respectively. This hardware while connected to the autopilot would receive desired motor speed in the form of PWM signal and would apply to the motor instantaneously. In other words, the interface between DC motors and autopilot is the ESC

http://www.imavs.org/

http://www.imavs.org/

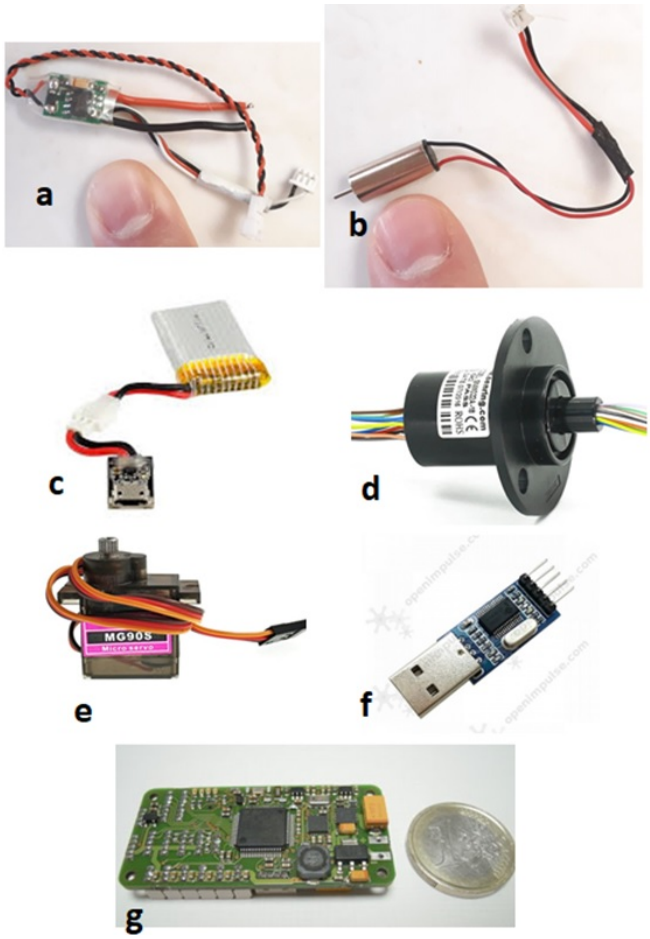


Figure 14: The important portions of hardware: a- speed control micro brush motor b- micro brush motor, c- battery d- slip ring, e- servo motor f- serial to USB converter, g- Apogee autopilot board version 1 .

hardware.

8 Designing and implementing an LQR controller

In designing an LQR control, the criteria for the performance of the system is introduced with a cost function of J, in which the controller tries to minimize it. This cost function is calculated as follows:

$$J = \int_0^\infty [x^T Q x + u^T R u] dt \tag{17}$$

In which Q is a positive semidefinite weighting matrix and can be normalized using the maximum values for states. The control weighting matrix R is a positive definite. Based on the actuator physics we should choose a suitable value for it. The R and Q matrices are very important regarding the errors. To have a successful LQR design, these matrices should be chosen very carefully. In this paper we use Bryson's rule. According to Bryson's rule, Q and R matrices are diagonal ones having elements on the main diagonal as inverse of the square of maximum values for states and control, respectively. The coefficient of the expression for $\dot{Q}_a$ in the equation of motion has the units as moment of inertia and therefore the expression $(-I_{ayy} - m_a(l_a^2 + l_a l_t \cos\Theta_a))\dot{Q}_a$ is equivalent to the moment that is exerted on the abdomen. We used $\dot{Q}_a$ as control input. The maximum values for pitch angle and pitch rate based on motor rpm and open-loop simulation results are determined as follows:

$$\begin{aligned} |\theta| &= |x_1| \leq 5^\circ \\ \left|\dot{\theta}\right| &= |x_2| \leq 20^\circ/\sec \end{aligned} \tag{18}$$

Based on the above values, Q can be determined as follows:

$$Q = \begin{bmatrix} \frac{1}{(5\pi/180)^2} & 0 \\ 0 & \frac{1}{(20\pi/180)^2} \end{bmatrix} \tag{19}$$

The control input is the product of lift force and the distance between suspension point to the effective point of lift force. Therefore, its maximum is proportional to maximum lift and can be obtained from equation (20):

$$-(-I_{ayy} - m_a(l_a^2 + l_a l_t \cos\Theta_a))\dot{Q}_a \leq -b_{FW} \times L_F + b_{BW} \times L_B \tag{20}$$

After substituting the values, maximum $\dot{Q}_a$ can be obtained from equation (21):

$$\dot{Q}_a \leq 6.072 \tag{21}$$

In the hardware in the loop, we fixed the abdomen and did not use $\dot{Q}_a$ as input. Therefore, the R matrix based on the maximum values for inputs can be obtained from equation (22):

$$R = \left[\frac{1}{(\Delta L_{\max})^2}\right] = \left[\frac{1}{(0.02 \times 9.81)^2}\right] \tag{22}$$

Now having matrices A, B, Q, and R, we can write Riccati equation as follows:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \tag{23}$$

We can obtain matrix P from the Riccati equation and then calculate optimal control gain (K) as in equation (24):

$$K = R^{-1}B^T P \tag{24}$$

After substituting the values for the gain matrix can be obtained as follows:

$$K = \begin{bmatrix} -2.0433 & -0.6586 \end{bmatrix} \tag{25}$$

The above controller gains were applied on linear model as well as to the nonlinear model and we can compare the responses.

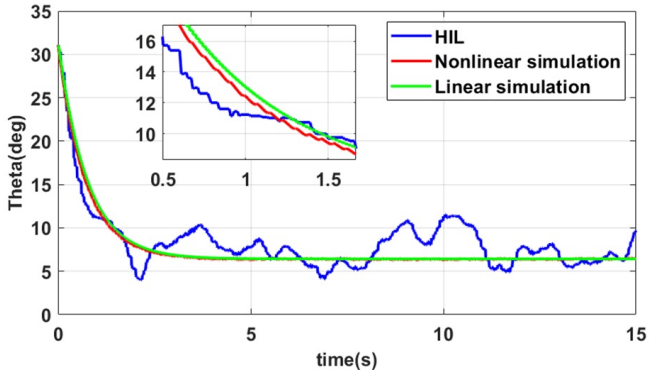


Figure 15: The closed loop time responses of for linear and nonlinear models.

The MAV is released from $\theta = 31°$ considering the equilibrium point of $\theta = 5°$. The time responses for the closed loop for θ are obtained as in figure 15 for linear and nonlinear models. By comparing the linear and nonlinear responses, it is observed that LQR gain which was obtained for linear model can work for nonlinear model as well.

9 Conclusions

In this paper, a new structure for the dragonfly-like MAV which had flapping wing with active rigid abdomen was proposed. Unlike previous research works, the abdomen of this MAV has no control role in flight. The nonlinear equations of motion were presented for two-body system in the presence of aerodynamic forces and the gearbox for the dragonfly-like MAV with active rigid abdomen. Then, the nonlinear dynamic equations of motion with nonlinear aerodynamics and nonlinear gearbox relationships were linearized. The good agreement between linear and nonlinear responses confirmed the validity of linearized equations. The positive role of abdomen motion in the open-loop as a vibration damper to create the left half-plane poles were discussed and analyzed. The LQR controller was designed for the linear system and the obtained gain was applied to both the linear and the nonlinear models. The closed loop responses were then compared. Moreover, hardware in the loop tests were performed using Paparazzi software and Apogee autopilot board. The results demonstrated appropriate robustness and accuracy of the designed controller as well as good agreement with numerical simulations in the presence of disturbances during real flight. In future works, the six degrees of freedom equations of motion similar to the longitudinal mode would be obtained. Furthermore, the controller design would be accomplished for the 6DOF case to study the abdomen role in stability.

References

[1] Anders Hedenström. Extreme endurance migration: what is the limit to non-stop flight? PLoS biology, 8(5):e1000362, 2010.

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

[3] Gareth D Padfield. Helicopter flight dynamics: including a treatment of tiltrotor aircraft. John Wiley & Sons, 2018.

[4] Hao Jie Zhu, Xue Guang Meng, and Mao Sun. Forward flight stability in a drone-fly. Scientific reports, 10(1):1–12, 2020.

[5] Shiba Biswal, Marc Mignolet, and Armando A Rodriguez. Modeling and control of flapping wing micro aerial vehicles. Bioinspiration & biomimetics, 14(2):026004, 2019.

[6] Sophie F Armanini, JV Caetano, GCHE De Croon, CC De Visser, and Max Mulder. Quasi-steady aerodynamic model of clap-and-fling flapping mav and validation using free-flight data. Bioinspiration & biomimetics, 11(4):046002, 2016.

[7] 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.

[8] Jeeva Jayakumar, Kei Senda, and Naoto Yokoyama. Control of pitch attitude by abdomen during forward flight of two-dimensional butterfly. Journal of Aircraft, 55(6):2327–2337, 2018.

[9] Joao V Caetano, CC De Visser, GCHE De Croon, B Remes, C De Wagter, J Verboom, and M Mulder. Linear aerodynamic model identification of a flapping wing mav based on flight test data. International Journal of Micro Air Vehicles, 5(4):273–286, 2013.

[10] Christophe De Wagter, Matěj Karásek, and Guido de Croon. Quad-thopter: tailless flapping wing robot with four pairs of wings. International Journal of Micro Air Vehicles, 10(3):244–253, 2018.

[11] Michael Bolender. Rigid multi-body equations-of-motion for flapping wing mavs using kane's equations. In AIAA Guidance, Navigation, and Control Conference, page 6158, 2009.

[12] Thomas R Kane and David A Levinson. Dynamics, theory and applications. McGraw Hill, 1985.

http://www.imavs.org/

[13] Mohammad Lashgari and Abolghasem Naghash. Modeling and linearization of longitudinal dynamics for a flapping wing micro aerial vehicle dragonfly-like with active rigid tail. Amirkabir Journal of Mechanical Engineering, 53(6):2–2, 2021.

10 nomenclature

a, b	link length of the mechanism
b_{BW} , b_{FW}	Distance between the effective lift force point for the back(front) wing and the suspension point
C_{mq}	pitching moment coefficient
D_F , D_B	Front(back) wing drag force
f_{motor}	MAV gearbox motor frequency
$F_{addedmass}$	Added mass forces
F_{circ}	Circulatory forces
$F_{inertial}$	Inertial forces
F_{visc}	Viscous forces
g	gravitational acceleration
h	second link length of the mechanism
I_{tyy} I_{ayy}	Thorax (abdomen) moment of inertia
L	Lift force
L_B L_F	The Back(Front) wing lift force
l_t l_a	Distance between CG of the thorax(abdomen) and the hinge
m_t m_a	Thorax(abdomen) mass
Q	Thorax pitch rate, the weighting matrix
Q_a	Abdomen angular velocity relative to thorax
$\dot{Q}_a$	Abdomen angular acceleration relative to thorax
R	control weighting matrix
U	MAV velocity component in x-direction of the body frame
u	Input control vector
W	MAV velocity component in z-direction of the body frame
α	Angle of attack
δ	Fixed angle of gearbox
ΔL	Differential lift
θ	Pitch angle
θ_i	Input angle of the gearbox
θ_a	Abdomen angle
ζ	Wing flap angle

Table 2: parameters of the dragonfly like MAV used in simulation.

http://www.imavs.org/

Design and Joint Control of a Conjoined Biplane and Quadrotor

S.Schröter*, E.J.J.Smeur, and B.D.W. Remes
Delft University of Technology, 2629 HS, the Netherlands

http://www.imavs.org/

ABSTRACT

Unmanned Aerial Vehicles (UAVs) have the potential to perform many different missions, some of which may require a large aircraft for endurance and a small aircraft for manoeuvrability in a building. This paper proposes a novel combination of a quadrotor and a hybrid biplane capable of joint hover, joint forward flight, and mid-air disassembly followed by separate flight. We investigate cooperative control strategies during joint flight that do not require any communication between the quadcopter and the biplane. This means that the two aircraft have their own independent control strategy based on their own sensors. Secondly, to avoid communication the biplane leads the flight and the goal for the quadrotor is to help in producing thrust and increasing stability. Three control strategies for the quadrotor are compared: a proportional angular rate damper, a proportional angular acceleration damper, and constant thrust without attitude control. Simulation and practical tests show that for intentional attitude changes of the biplane, the quadrotor rate- and angular acceleration damper strategies lead to a small performance degradation. However, the angular rate damper strategy for disturbance rejection has the lowest roll angle error and requires the smallest input command. The in-flight release is successfully tested in joint hover up to a forward pitch angle of -18 [deg].

1 INTRODUCTION

Unmanned Air Vehicles (UAVs) have increased in popularity and can serve various purposes, ranging from inspection of structures to traffic surveillance, and each type of UAV has its own distinguishing properties. Fixed-wing aircraft are known for their endurance and efficiency, but they require a constant horizontal speed to stay in the air. Multirotors are more agile and are capable of hover, but they lack endurance. A hybrid aircraft can hover and has wings for efficient forward flight, combining the best of both worlds. However, a hybrid aircraft with large wings is not well-equipped to maneuver inside a building. This poses a problem for missions where the goal is to fly inside a building after a long-distance transit flight.

One solution could be to drop a smaller UAV out of a bigger one. In-air deployment of a fixed-wing from a quadrotor has been shown by Boeing[1]. Voskuijl et al. investigated morphing UAVs being dropped as armaments out of (military) airplanes [1]. A downside of this approach is that the smaller UAV is carried around as dead weight, and does not contribute to the propulsion until it is deployed. This could lead to over-dimensioning of the carrier aircraft. Cooperative flight is somewhat similar to airborne docking [2], formation flight with communication [3], and without communication [4]. However, in these works the individual UAVs are typically not rigidly attached and can maneuver independently to some degree.

Examples of cooperative flight with modular joint airframes are the Modquad [5] and the Distributed Flight Array [6]. Both airframes are capable of assembling in-flight, and the latest version of the Modquad is even capable of in-flight disassembly [7]. However, all aircraft used in this research have the same size and function, without a focus on combining different types of UAVs to obtain better endurance and flexibility in operation.

In this paper, we propose a combination of a quadrotor and a hybrid aircraft with a fixed wing, that can fly together, both contributing to stabilization and propulsion, and can disassemble in-flight into a fully functional quadcopter and hybrid aircraft. Figure 1 graphically shows the various stages in which the joint structure would fly.

For a UAV's attitude and trajectory control specifically, many options exist. The most popular method is PID control [8], but also model-based controllers exist for UAVs [9]. For hybrid aircraft, Incremental Nonlinear Dynamic Inversion (INDI) has proven to be very effective [10]. Especially in the hover phase, hybrid aircraft are extremely susceptible to external disturbances [11, 12], and INDI has better disturbance rejection compared to PID [13].

Different forms of communication have been used in all forms of cooperative flight between different aircraft, but this has proven to have its challenges, such as time delays, false information and noise. Wired communication between the aircraft would be less of a liability, but a way to work around all these problems is to avoid any communication altogether. That is why a control strategy is proposed in this paper where

*Email address: shawn@schroter.info

[1] https://www.boeing.com/features/2016/09/catch-and-release-flares-09-16.page

both aircraft have no communication with each other. This way, the joint structure becomes more reliable, which is important for long-distance operations.

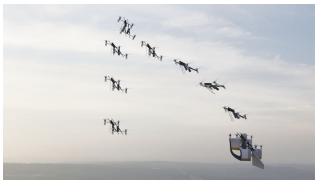
Two main questions will be dealt with: how can the joint vehicle be controlled and how can the two aircraft be disassembled in mid-air?



(a) Joint Hover



(b) Joint Forward flight



(c) In-flight release

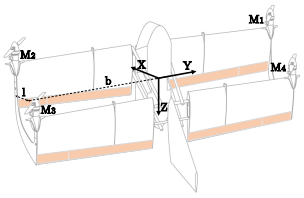


(d) The quadrotor manoeuvring in a building, while the biplane loiters (in forward flight) as a data relay station

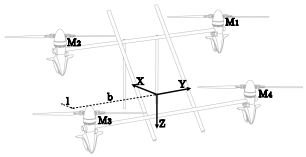
Figure 1: Different phases of the joint structure's activities

2 Joint structure design

This section will cover three main parts of the structure in more detail; The hybrid biplane, the quadrotor and the release mechanism.



(a) Hybrid biplane



(b) Quadrotor

Figure 2: Schematic drawing of the hybrid biplane and quadrotor with axis definitions. The control surface are accentuated in light orange. M_i illustrates the ith rotor actuators.

2.1 *Hybrid biplane*

The biplane is a tailsitter hybrid aircraft. For this aircraft, the Nederdrone formed the basis of the biplane design [10]. The biplane has eight mounting points for rotors and four control surfaces. The four outer mounting points are fixed, and the four inner mounting points are part of the release mechanism. The control surfaces control the airflow around the wings, providing moments around the Y- and Z-axes. In the hover phase, this airflow is created by the rotors. Table 1 presents an overview of the different parts of the hybrid biplane. The reference frame is defined in hover state, as shown in Figure 2a. Forward flight would mean a $-90[deg]$ pitch angle.

Type of Hardware	Brand	Item
Motor	T-Motor	MN3510
Radio Control link	TBS	Crossfire nano
Telemetry link	Herelink	Herelink
Electronic Speed Controller	T-Motor	f45A 32_bit
Propeller	T-motor	MF1302
Flight controller	Holybro	Pixhawk4
Battery	Extron	2x 6s 4.5 Ah

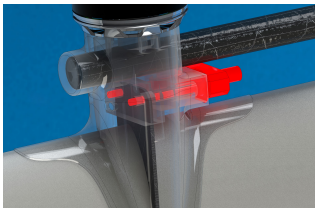
Table 1: Different components of the biplane

2.2 *Quadrotor*

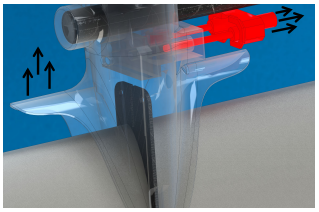
The design of the quadrotor is derived from the dimensions of the biplane. This means that on the four inner mounting points of the biplane the four rotors of the quadrotor are placed. Four hollow carbon rods connect the four rotors. The rods are placed in such a way that the quadrotor would be able to release without being obstructed by the biplane. Furthermore, two solid carbon guiding support a more consistent release and serve as landing legs. A schematic overview of the quadrotor is shown in Figure 2b. Figure 1c shows the release if the biplane is in hover. The powertrain is similar to the biplane. One difference is that the quadrotor uses two 3s 4500 mAh batteries, connected in series.

2.3 *Release mechanism*

The release mechanism is mounted just underneath the motors of the quadrotor. At each of the four motor locations, two copper pins fix the quadrotor to the biplane. The release system is a slider-crank mechanism powered by a servo. Since the quadrotor needs to drastically change its method of control once it detaches from the biplane, it is given control over the release mechanism.



(a) Closed position - The pin is locked in the carbon plate.



(b) Lift-off of the quadrotor

Figure 3: Closed position and release of the RC servo Release Mechanism

Figure 3 graphically shows how the release system enables the quadrotor to separate from the biplane. Once the pins are out, the motor mount slides in the direction of the

http://www.imavs.org/

thrust force. A thrust difference between the biplane and the quadrotor then leads to separation. The guiding tubes of the quadrotor help the thrust force of the quadrotor to stay approximately opposite to the weight force of the biplane.

3 MODEL STRUCTURE

This section will explain how the aircraft models are defined to control the joint structure properly.

3.1 Control law

Both aircraft will be flying in three different flight phases: joint hover, joint forward flight, and separate hover. A control law for both hover phases will need to be designed first, since joint forward flight can only be achieved after the joint hover phase is properly controlled.

For the flight control strategy, INDI is chosen as a starting point. INDI has proven to perform very well for tailsitter hybrid aircraft. These aircraft have complex aerodynamics during the different flight phases and are very sensitive to wind gusts in hover. INDI treats modeling errors as disturbances, to which it has good rejection capabilities [10, 13]. The following gives a brief overview of the control method, but the interested reader is referred to [13]. For both aircraft the following angular momentum equation holds:

$$\begin{aligned} M &= I_v\dot{\Omega} + \Omega \times I_v\Omega \\ &= M_a(\Omega, v) + M_c(\omega) + M_r(\omega, \dot{\omega}, \Omega), \end{aligned} \tag{1}$$

where M is the total moment, I_v is the inertia around the rotational axis of the aircraft, ω is the angular rate of the propellers around the body Z-axis and $\dot{\omega}$ is the angular acceleration of the propellers around the body Z-axis. M_a is the moment due to aerodynamics, M_c is the moment due to the controls, M_r is the moment due to the gyroscopic effect of the rotors and Ω is the angular rotation vector.

Eq. 1 can be used to derive the general control law of INDI, described as:

$$\omega_c = \omega_f + (G_1 + G_2)^{+}(v - \dot{\Omega}_f + G_2 z^{-1}(\omega_c - \omega_f)), \tag{2}$$

where ω_c is the new motor command, ω_f is the filtered motor command of the previous iteration, G_1 describes the control effectiveness of the actuators, G_2 describes the propeller inertia effect on the Z-axis, $\dot{\Omega}_f$ is the filtered measured angular acceleration, and v is the virtual control input. This virtual control is defined as the reference angular acceleration created by a PD controller. A full derivation is given in [14].

3.2 Control authority analysis

To analyse how the two aircraft compare considering control authority, the control moment from Eq. 1 is used. Per axis, the control moment, M_c, is defined for a quadrotor as [15]:

$$M_c = \begin{bmatrix} -bk_1 & bk_1 & bk_1 & -bk_1 \\ lk_1 & lk_1 & -lk_1 & -lk_1 \\ k_2 & -k_2 & k_2 & -k_2 \end{bmatrix} \omega^2, \tag{3}$$

where b is the lateral distance from the Center of Gravity (CG) to the rotors, l is the longitudinal distance between the CG and the rotors, k_1 is the force constant of the rotors, k_2 is the moment constant of the rotors, and ω is the angular rate vector of the rotors.

From Eq. 3 the control effectiveness of the different actuators can be derived. The actuator force constants are the same for the biplane and the quadrotor, since the same hardware is used for both aircraft. If the control surfaces were negated, both the biplane and the quadrotor separately can be seen as two quadrotors, where Eq. 3 holds. Table 2 shows an overview of b and l for the biplane and the quadrotor.

	b	l
Biplane	0.74 [m]	0.11 [m]
Quadrotor	0.32 [m]	0.11 [m]

Table 2: lateral and longitudinal distances from rotor actuators to CG for the biplane and the quadrotor.

By comparing b and l in Table 2, it can be observed that it is only around the roll axis the biplane has more control authority than the quadrotor from the propellers alone. Furthermore, for pitch and yaw, the control surfaces of the biplane aid in control. In conclusion, in all rotational directions the biplane has more control authority than the quadrotor.

4 CONTROL STRATEGIES FOR THE QUADROTOR IN JOINT HOVER

One of the main difficulties of the project is the lack of intercommunication. It is deemed infeasible to let both UAVs track a reference model due to the difficulty of synchronisation and (different) sensor errors. As the UAV with the most control authority, the biplane is provided with a reference model to track.

The goal is to have the quadrotor improve the overall performance of the joint structure. Firstly, tracking performance and disturbance rejection behaviour is preferred for tailsitter platforms like the biplane. Secondly, the least amount of energy should be required to perform flight. Thirdly, the input commands should be as small as possible. Less required input command means that the actuators are further away from their saturation point, giving the actuators more room for extra manoeuvring.

The simplest strategy for the quadrotor is to provide a constant thrust, without creating any control moments. This makes the biplane fully responsible for the attitude control and adjusting the total amount of thrust. This control strategy will be referred to as *constant thrust*.

Due to the lack of communication, a major challenge for the quadrotor is to distinguish the difference between intended change in attitude or an external disturbance changing the attitude. Intended behaviour is created by changing reference signals, stemming from an outer-loop position controller or manual input from an RC controller. Unintended

http://www.imavs.org/

behaviour is usually the result of external forces acting on the platform, for instance due to wind gusts. We will investigate if the quadrotor can improve the overall performance by resisting rotations, even though that means resisting intended rotations as well.

The next two investigated strategies are the *angular rate damper* and *angular acceleration damper*. These strategies will resist angular rates and angular accelerations respectively. The type of control is proportional control, as adding an integrator could lead to a steady state of opposing moments of the quadrotor and biplane.

Figure 4 shows the block diagrams for the three strategies. With the damper strategies, both the biplane and the quadrotor will detect a disturbance and will try to steer against this, though the quadrotor will also detect the intended behaviour of the biplane as a disturbance.

4.1 Control group for base reference

In order to compare the effect of the different quadrotor control strategies, performance has to be tested against a base reference. This base reference is one INDI attitude controller directly controlling *all eight rotor actuators and the control surfaces*. This makes the control group physically different from the three investigated control strategies for the quadrotor. For this strategy to work, the motors of the quadrotor are directly wired to the biplane. In the next sections this base reference is referred to as the *control group*.

4.2 Stability analysis

Figure 5 shows a Nichols plot derived for the different control strategies. This plot is created for the entire joint structure controller, as displayed in Figure 4. The goal is to have the system's frequency response be as far from the critical point, the red cross, in the middle. The vertical distance from the system's frequency response to the critical point illustrates the gain margin, and the horizontal distance defines the phase margin. It becomes clear that the angular acceleration damper strategy is the least robust, closely followed by the constant thrust strategy. The control group overlaps again with the constant thrust strategy. The angular rate damper is the most robust, showing better gain and phase margins than the control group.

5 PRACTICAL VERIFICATION

This section is focused on the joint structure's release mechanism and the test sequence for the control strategies. The test sequence has been simulated in Matlab prior to the practical flights, but will not be further mentioned due to the page restrictions for this paper.

5.1 Release Mechanism

The joint structure was built as per description mentioned in Section 2. The sequence of release is shown in Figure 6. It is important to note that the quadrotor does not control its attitude during the release. This is because the motor mount of the quadrotor has to slide off a carbon plate, as shown in Figure 3, and if a moment is applied this causes friction. If the quadrotor was using attitude control, or even one of the damper strategies, it could steer against the biplane and through the generated moment obstruct proper detachment.

On average, the quadrotor was clear of the biplane within $0.39\ [sec]$. The total delay time before activation of the attitude controller was iteratively set to $0.5[sec]$. This left enough time for the quadrotor to be clear of the biplane and not be in free flight for too long without active attitude control. The set-up was tested with varying forward pitch angles up to $\theta = -18\ [deg]$. At 70% thrust, the quadrotor consistently released well.

The release system was also tested outdoors. The biplane was controlled by a separate pilot using an RC controller. At the moment of release, the goal was to have all the angles be as close to as possible to $[\phi, \theta, \psi] = [0, 0, 0]$ degrees. Figure 7 shows the three phases of the in-flight release. Next, both platforms were proven to be capable of separate flight. In the case of the biplane, hover flight was carried out with the four outer rotors and the control surfaces.

The biplane was tuned for flight with the quadrotor attached. This means that, for the calculated control effectiveness, the inertia of the joint structure is taken into account, including the mass of the quadrotor. When the quadrotor releases, the actual control effectiveness increases due to the decreased inertia. This effectively leads to a gain higher than 1 in the control system. According to Figure 5, there is some degree of robustness to such a gain change. In practice the flight performance of the biplane without the quadrotor proved adequate. One reason could be that the weight of the quadrotor sits close to the CG, which only leads to a small change in inertia once the quadrotor detaches.

5.2 Control strategies - test setup

Two experiments were performed: one to test the tracking performance and one to test disturbance rejection. For the first experiment, the intentional step input was initialised via the RC controller. The step was set to $\phi = 18\ [deg]$ in both positive and negative roll angles.

The experiments were performed indoors to avoid any influence of wind, in one sitting. In all cases, the thrust level for the quadrotor was set to 70%. Table 3 shows the mean throttle levels for the biplane for the various configurations.

Mean Throttle level	Biplane	Quadrotor
control group	62.13 [%]	-
ang. rate damper	53.90 [%]	70 [%]
constant thrust	53.90 [%]	70 [%]
ang. acc. damper	54.21 [%]	70 [%]
separate flight	72.33 [%]	40 [%]

Table 3: throttle levels for different configurations

In order to create a repeatable and consistent step disturbance a weight was dropped from the joint structure. Two identical weights of $672\ [g]$ each were put on the sides of the

http://www.imavs.org/

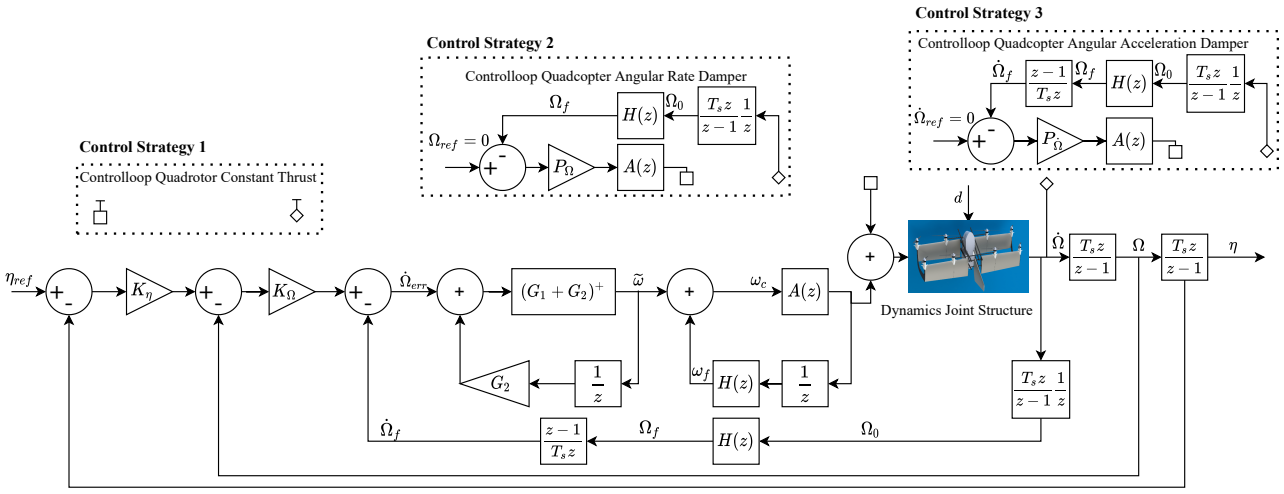


Figure 4: Three control strategies for the quadrotor. The lower section of the loop shows the INDI cascaded attitude controller of the biplane. The area within the dotted line resembles the control loop for the quadrotor.

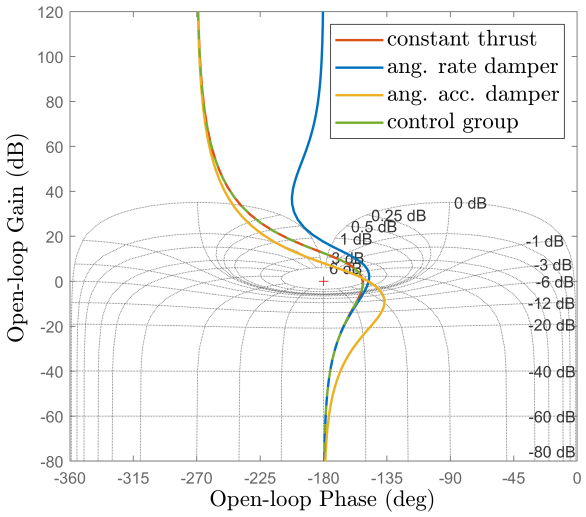


Figure 5: Nichols plot for three strategies. Constant thrust and the control group overlap. The angular rate damper strategy should provide the best stability.

biplane, one weight on each side. This created a net zero moment on the roll axis, not changing the CG, in the beginning of the test. After the release of one weight, the CG was shifted towards the weight that was still attached, resulting in a constant roll moment. This step disturbance is comparable to [16], but Smeur et al. added a weight to the aircraft.

An effort was made to keep the flight controller software similar for the different controllers to the greatest possible extent. The actuator dynamics, the filtering, and the sensor fusion all took place in a similar fashion for the quadrotor as well as the biplane. The main differences were the measurements and the actual control law.

5.3 *Control strategies - test results*

Figure 8 illustrates the overall result of all the roll angle step input tests. Figure 8a and 8b show the mean response and input command respectively of 6 repetitions for the angular rate and angular acceleration damper strategies, 5 repetitions for the control group and 3 repetitions for the constant thrust strategy. Table 4 and 5 show relevant parameters of the step input and disturbance.

Figure 8a shows that the step input tests give a very similar response for all the strategies. Overshoot for all of the strategies does not go above 5% of the step angle. The new steady-state roll angle could not be maintained for a long time, due to a lack of space in the hangar. This also explains why the initial roll angle does not show a good steady-state value for some responses.

The increase of the roll command after $t = 2\ [sec]$ can be explained by the fact that the joint structure needs to put in a constant roll offset to keep the roll angle at $\phi = 18\ [deg]$. This is referred to as a head-up moment, due to the flapping movement of the rotor [17].

Where the difference between strategies for the step responses is small, the difference for disturbance rejection is more significant. Table 5 shows that the control group has the lowest ϕ_{err}^{max} and the smallest required input command, as expected. Comparing this base reference strategy to the other strategies, we find that the constant thrust would require 19% more peak input command to counteract the disturbance. The angular acceleration damper strategy performs even worse at 27% more required peak input command. The angular rate

http://www.imavs.org/

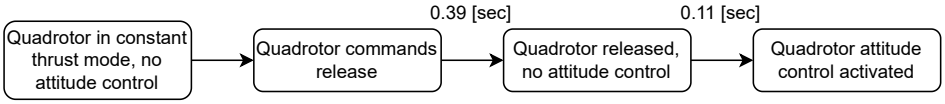
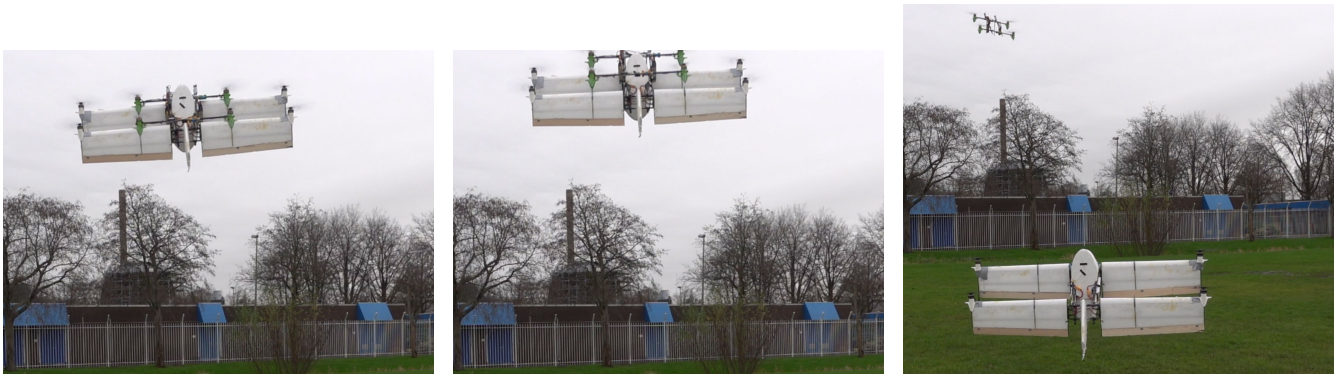


Figure 6: Flowchart of the release mechanism



(a) Joint hover; Biplane in attitude INDI and quadrotor constant 70% thrust.

(b) Release mechanism engaged; The quadrotor has no attitude control.

(c) Separate flight; Both aircraft are in attitude control.

Figure 7: Field release test

damper strategy performed the best, as it only required 11% more peak input command compared to the control group. The same order of performance can be stated for ϕ_{err}^{max} and the total energy required.

At this point it should be borne in mind that the control group for this study acted as a base reference only, not as a feasible option. Of the options that were deemed feasible, the angular rate damper strategy proved to be the best.

6 CONCLUSION

This paper proposes a combined aerial system consisting of a biplane and quadrotor attached together, that can hover cooperatively, and disassemble in-flight followed by separate flight. In-flight release worked consistently and is tested up to a forward pitch angle of -18 [deg]. Given the fact that no intercommunication is available, the angular rate damper control strategy for the quadrotor helps with disturbance rejection, while marginally affecting performance in terms of intended behaviour. Compared to the control group strategy, no active attitude control would result in 19% extra command. With the rate damper strategy active, this is reduced to 11% extra input command needed. Also, the maximum angle that the joint platform would reach due to the disturbance and the total amount of energy required is reduced with the active damper strategy. The angular acceleration damper strategy performed significantly worse for both the extra input command and the maximum angle.

7 DISCUSSION, FURTHER RESEARCH

The system is designed to also be able to fly in fast forward flight, but this has not been tested within the scope of this paper. Future work could investigate the stability and tracking performance during transition and forward flight as well. Additionally, future work could include examining how the quadrotor could be given more knowledge of the reference trajectory. This could be done with geofencing to let the quadrotor know what phase of flight the joint structure is in. One could raise the question if it is worth it to avoid communication between the drones, as the results show that it does come at the cost of a small degradation in performance. Additionally, without communication the biplane may reach saturation limits quicker, as it has lower control authority compared to the case where it can send a command to the quadrotor. Communication would also allow for the quadrotor to adjust the thrust setting to the most efficient level, instead of the constant value that it had in this paper.

REFERENCES

[1] Mark Voskuijl, Muhammad R Said, Jaspreet Pandher, Michel J van Tooren, and Blin Richards. In-flight deployment of morphing uavs–a method to analyze dynamic stability, controllability and loads. In *AIAA Aviation 2019 Forum*, page 3126, 2019.

[2] Daniel Briggs Wilson, Ali Göktogan, and Salah Sukkarieh. Guidance and navigation for uav airborne docking. In *Robotics: Science and Systems*, volume 3, 2015.

[3] Ruibin Xue and Gaohua Cai. Formation flight control of multi-uav system with communication con-

http://www.imavs.org/

	Settling time	Max. input command	Energy required for t=[1-3.5]
control group	0.89 [sec]	23.62 [%]	8.00 [%· s]
ang. rate damper	0.84 [sec]	22.76 [%]	8.18 [%· s]
constant thrust	0.84 [sec]	23.56 [%]	8.96 [%· s]
ang. acc. damper	0.81 [sec]	23.73 [%]	10.31 [%· s]

Table 4: Practical results for the different strategies for intended step inputs.

	Max. ϕ_{error}	Max. input command	Relative change	Energy required for t=[0-2.25]
control group	5.19 [deg]	26.64 [%]	100 [%]	31.44 [%· s]
ang. rate damper	5.79 [deg]	29.62 [%]	111 [%]	38.78 [%· s]
constant thrust	6.29 [deg]	31.67 [%]	119 [%]	40.38 [%· s]
ang. acc. damper	6.92 [deg]	33.84 [%]	127 [%]	42.62 [%· s]

Table 5: Practical results for the different strategies for disturbance rejection. For the input command, the relative change is given with respect to the base reference of the control group.

http://www.imavs.org/

straints. *Journal of aerospace technology and management*, 8(2):203–210, 2016.

[4] Mohammad A Dehghani and Mohammad B Menhaj. Communication free leader-follower formation control of unmanned aircraft systems. *Robotics and Autonomous Systems*, 80:69–75, 2016.

[5] David Saldana, Bruno Gabrich, Guanrui Li, Mark Yim, and Vijay Kumar. Modquad: The flying modular structure that self-assembles in midair. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 691–698. IEEE, 2018.

[6] Raymond Oung and Raffaello D'Andrea. The distributed flight array: Design, implementation, and analysis of a modular vertical take-off and landing vehicle. *The International Journal of Robotics Research*, 33(3):375–400, 2014.

[7] David Saldana, Parakh M Gupta, and Vijay Kumar. Design and control of aerial modules for inflight self-disassembly. *IEEE Robotics and Automation Letters*, 4(4):3410–3417, 2019.

[8] Atheer L. Salih, M. Moghavvemi, Haider A. F. Mohamed, and Khalaf Sallom Gaeid. Modelling and pid controller design for a quadrotor unmanned air vehicle. In *2010 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, volume 1, pages 1–5, 2010.

[9] Hongwei Mo and Ghulam Farid. Nonlinear and adaptive intelligent control techniques for quadrotor uav–a survey. *Asian Journal of Control*, 21(2):989–1008, 2019.

[10] Christophe De Wagter, Bart Remes, Ewoud Smeur, Freek van Tienen, Rick Ruijsink, Kevin van Hecke, and Erik van der Horst. The nederdrone: A hybrid lift, hybrid energy hydrogen uav. *international journal of hydrogen energy*, 46(29):16003–16018, 2021.

[11] Hang Zhang, Bifeng Song, Haifeng Wang, and Jianlin Xuan. A method for evaluating the wind disturbance rejection capability of a hybrid uav in the quadrotor mode. *International Journal of Micro Air Vehicles*, 11:1756829319869647, 2019.

[12] Bart Theys, Cyriel Notteboom, Menno Hochstenbach, and Joris De Schutter. Design and control of an unmanned aerial vehicle for autonomous parcel delivery with transition from vertical take-off to forward flight. *International Journal of Micro Air Vehicles*, 7(4):395–405, 2015.

[13] Ewoud JJ Smeur, Guido CHE de Croon, and Qiping Chu. Cascaded incremental nonlinear dynamic inversion for mav disturbance rejection. *Control Engineering Practice*, 73:79–90, 2018.

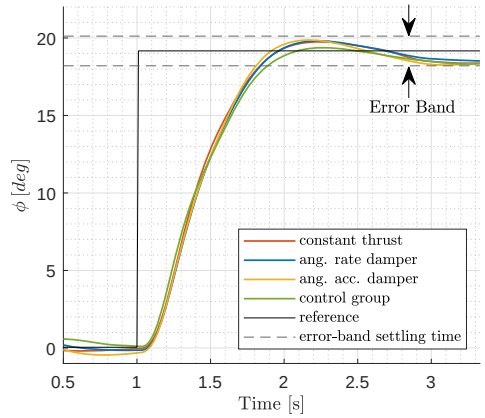
[14] Ewoud JJ Smeur, Qiping Chu, and Guido CHE de Croon. Adaptive incremental nonlinear dynamic inversion for attitude control of micro air vehicles. *Journal of Guidance, Control, and Dynamics*, 39(3):450–461, 2016.

[15] Robert Mahony, Vijay Kumar, and Peter Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics and Automation magazine*, 19(3):20–32, 2012.

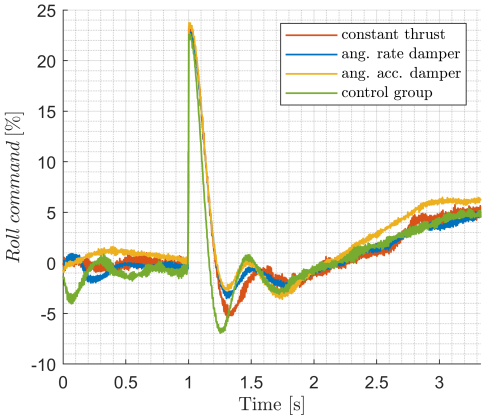
[16] EJJ Smeur. *Incremental Control of Hybrid Micro Air Vehicles*. PhD thesis, Delft University of Technology, 2018.

[17] Hikaru Otsuka, Daisuke Sasaki, and Keiji Nagatani. Reduction of the head-up pitching moment of small quadrotor unmanned aerial vehicles in uniform flow. *International Journal of Micro Air Vehicles*, 10(1):85–105, 2018.

http://www.imavs.org/

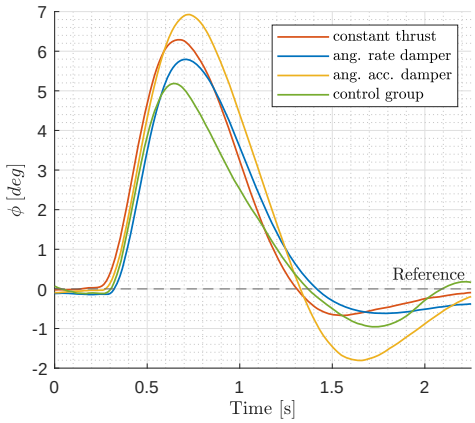


(a) Roll angle after intended step input with a 5 % error band.

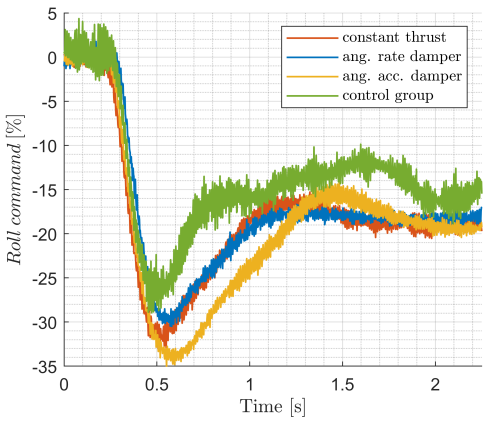


(b) Roll input command during intended step input. After 2 [sec] the input gradually increases due as the UAV picks up speed.

Figure 8: Intended step input in the Hover phase.



(a) Roll angle for disturbance rejection.



(b) Roll input command for disturbance rejection.

Figure 9: Intended step disturbance rejection in the Hover phase.

Total Energy Control System Design for Longitudinal Dynamics of Complex Articulated Ornithopter

Woongtaek Oh, Inrae Kim, Seungkeun Kim*, and Jinyoung Suk
Chungnam National University, Repulic of Korea

http://www.imavs.org/

ABSTRACT

This paper describes the design of total energy control system(TECS) for longitudinal flight of a complex articulated ornithopter. Aerodynamic coefficients are acquired with an analysis tool based on the Vortex Lattice Method. Equations of motion reflecting the flapping-wing effect are designed, and available control inputs are selected. A flight controller using these control inputs is designed, and simulation results are analyzed.

1 INTRODUCTION

TECS controls aircraft's thrust and pitch angle based on the total energy demand and the total energy distribution error calculated by flight altitude and speed. The thrust command equals the total energy to the desired total energy, and the pitch angle command reduces distribution error of the kinetic and the potential energy. In general fixed-wing aircraft, the propulsion equipment and the elevator are operated by the thrust and pitch angle command. A. A. Lambregts introduced autopilot using this principle [1].

Jang and Han analyzed the PX4 project, that is an open platform of the autopilot hardware and software for developing drones [2]. This platform is popular and used in many research institutes because of its compatibility with various aircraft. The TECS principle is used in the PX4 to control fixed-wing aircraft. So, using this principle can help the improvement of accessibility to develop an ornithopter system.

The ornithopter does not have propulsion equipment and generates thrust and lift by flapping the wing. To control this system, many research about the analysis of flapping-wing effects have been conducted. D. Kumar *et al.* did a wind tunnel experiment to analyze the flapping frequency and stroke-ratio effects of a single articulated ornithopter [3]. And Wolfgang Send *et al.* analyzed active wing twist control effects of a complex articulated ornithopter [4].

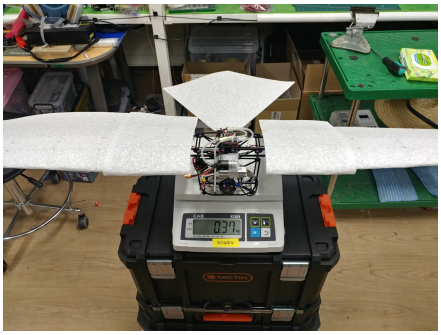
This paper contains research about the longitudinal simulator design of the complex articulated ornithopter. First, available control inputs are selected, and their effects are considered. Then, desired flight altitude and speed are set, and the TECS controller is designed with the selected control inputs. Finally, the simulation results are analyzed.

*Email address: skim78@cnu.ac.kr

2 EQUATIONS OF MOTION MODELING

2.1 Aerodynamic analysis

Chungnam National University has researched the articulated ornithopter system. Kim *et al.* designed and fabricated a complex articulated ornithopter named USGull-mini, and did flight test in 2018 [5]. The research continues recently, and USGull-mini flight performance improvements have been carried out. The specification is shown in Table 1.



Wing span	1,450 mm
Weight	370 g
Max. flapping frequency	4 Hz

Table 1: USGull-mini specification

The inner and outer wing folding angles are changed according to the flapping-motion affecting the aerodynamic coefficients change. Both wing's folding angles should be considered separately in the complex articulated structure unlike the single articulated. The flapping-wing motion states can be divided into 12 steps based on each wing's folding angle increase/decrease occurrence as shown in Figure 1.

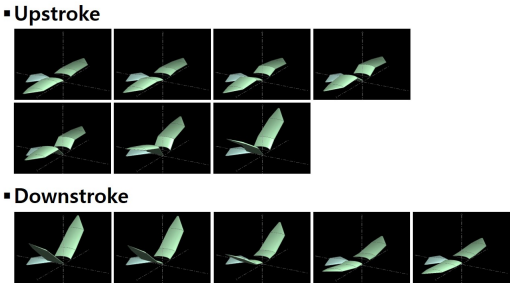


Figure 1: The flapping-wing motion of USGull-mini

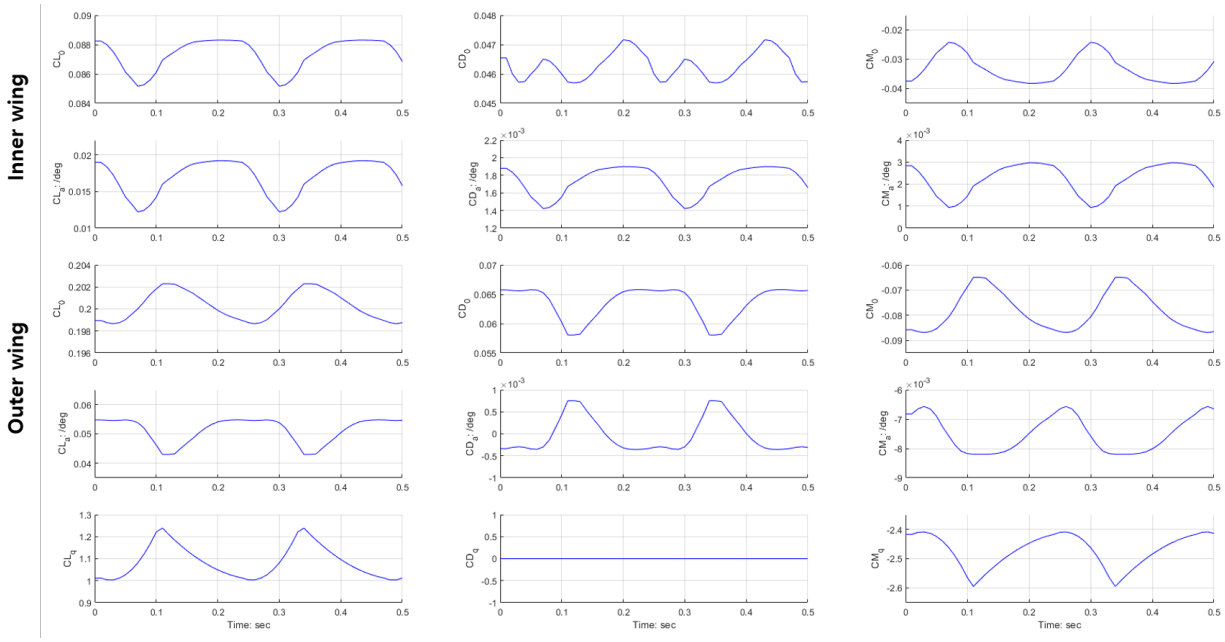


Figure 2: Aerodynamic coefficient change for two periods of flapping motion

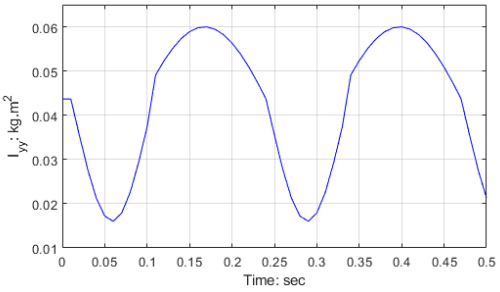


Figure 3: Moment of inertia change for two periods of flapping motion

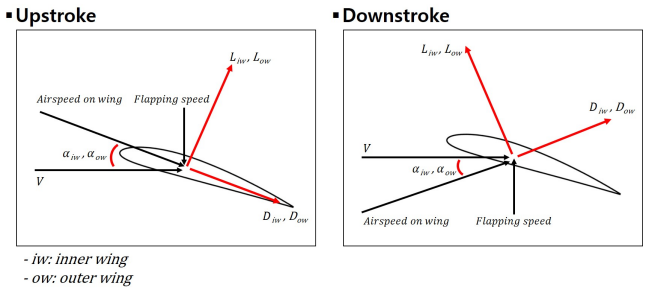


Figure 4: Lift and drag geometry during upstroke and downstroke

XFLR5 is an analysis tool based on the Vortex Lattice Method (VLM) suitable for aircraft at low Reynolds number. HUA Zhao-min *et al*. used XFLR5 to compare aerodynamic efficiencies of two-section and three-section flapping-wing mechanism [6]. This research showed that XFLR5 can analyze the articulated ornithopter. The lift, drag, and moment coefficients are acquired by its aerodynamic analysis. Figure 2 represents the flapping effects on the coefficients at a condition of flapping frequency 4Hz. In addition to the aerodynamic coefficient, the moment of inertia changes according to the flapping-motion, as shown in Figure 3.

2.2 *Equations of motion modeling*

Flapping-wing motion generates an airspeed along the z-axis of the aircraft, and it causes the angle of attack change on the wings. As a result, the lift's x-axis component is replaced by thrust. Figure 4 represents these effects, and it is needed to design equations of motion. The equations of motion related with the longitudinal axis are given in equation (1)-(4).

$$\dot{V} = (L_{iw}sin\alpha_{iw} - D_{iw}cos\alpha_{iw} + L_{ow}sin\alpha_{ow} - D_{ow}cos\alpha_{ow} - D_{tw} - D_q - mgsin\gamma)/m \quad (1)$$

$$\dot{\alpha} = (-L_{iw}cos\alpha_{iw} - D_{iw}sin\alpha_{iw})/(mV_{iw}) + (-L_{ow}cos\alpha_{ow} - D_{ow}sin\alpha_{ow})/(mV_{ow}) + (-L_{tw} - L_q + mgcos\gamma)/(mV) + q \quad (2)$$

$$\dot{q} = (M_{iw} + M_{ow} + M_{tw} + M_q)/I_{yy} \quad (3)$$

$$\dot{\theta} = q \quad (4)$$

where $L = \bar{q}SC_L$, $D = \bar{q}SC_D$, and $M = \bar{q}ScC_M$. L and D denote lift and drag forces, M is pitch moment. And V is the airspeed, α is the angle of attack. q is the pitch rate and θ is

http://www.imavs.org/

the pitch angle. γ and g are flight path angle and gravity constant. m, S, c, and I_{yy} are physical properties representing the mass, the wing surface, the mean chord length, and the moment of inertia. iw, ow, and tw are abbreviations meaning inner wing, outer wing, and tail wing. And $\overline{q}$ is the dynamic pressure. Finally, C_L, C_D, and C_M are the aerodynamic coefficients of lift, drag, and pitch moment.

3 TOTAL ENERGY CONTROL SYSTEM DESIGN

3.1 Control input analysis

A simulation for control input analysis is configured and its contents are as follows. The aircraft takes off from the ground and climbs to an altitude of 30 m. After reaching the desired altitude, the aircraft cruises maintaining it. There is no desired flight speed, but it has the limitation of plus/minus 20 ° in the pitch angle command. Flapping frequency, stroke-ratio, and active wing twist angle control are selected as control input candidates. Stroke-ratio is expressed as downstroke speed divided by upstroke speed, and active wing twist angle is controlled by an additional actuator mounted on the wingtip. The thrust and lift effects from flapping-motion according to the changes of these candidates are shown in Figure 5.

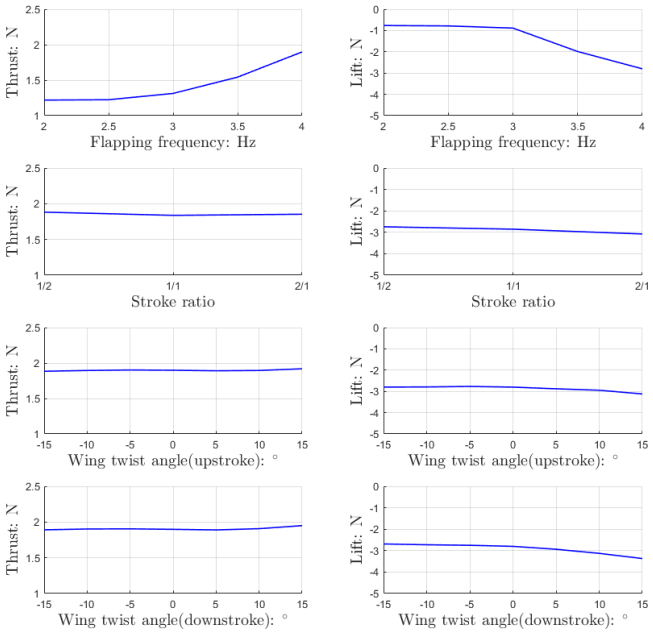


Figure 5: Control input effects

Each control input's effects are as follows. The increase in flapping frequency causes an increase in the thrust and a decrease in the lift. And the decrease in stroke-ratio causes an increase in the thrust and the lift. Finally, the increase in wing twist angle in the upstroke state causes an increase in the thrust and a decrease in the lift, but the increase in the thrust is negligible, and the same trends are displayed in the downstroke state. These analysis results are used to design the TECS controller.

3.2 Total energy control system design

TECS controller consists of two loops, total energy control and total energy balance control. The total energy control is based on the equation expressed as the kinetic and the potential energy sum. And the total energy balance control is based on the equation expressed as the difference between the kinetic and the potential energy. It is represented as follows.

$$E_T = \frac{1}{2}mV^2 + mgh \tag{5}$$

where E_T is the total energy, the energy rate can be derived as:

$$\dot{E}_T = mV\dot{V} + mg\dot{h}. \tag{6}$$

The energy balance E_B and the energy balance rate $\dot{E}_B$ can be defined in the same way.

$$E_B = -\frac{1}{2}mV^2 + mgh \tag{7}$$

$$\dot{E}_B = -mV\dot{V} + mg\dot{h} \tag{8}$$

The TECS controller determines thrust command using the total energy control loop and pitch angle command using the total energy balance control loop. In general fixed-wing aircraft, the thrust and pitch angle command affect propulsion equipment and elevator operations. The ornithopter also has an elevator, but it does not have propulsion equipment to generate thrust. So, the TECS controller should be modified to be suitable for this system, like the block diagram in Figure 7. The propulsion equipment control is replaced by flapping frequency and active wing twist angle control. Control concept representing the above is as follows.

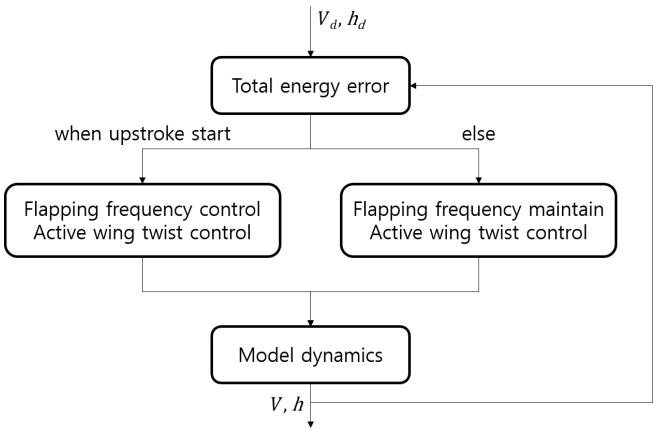


Figure 6: Total energy control loop concept

http://www.imavs.org/

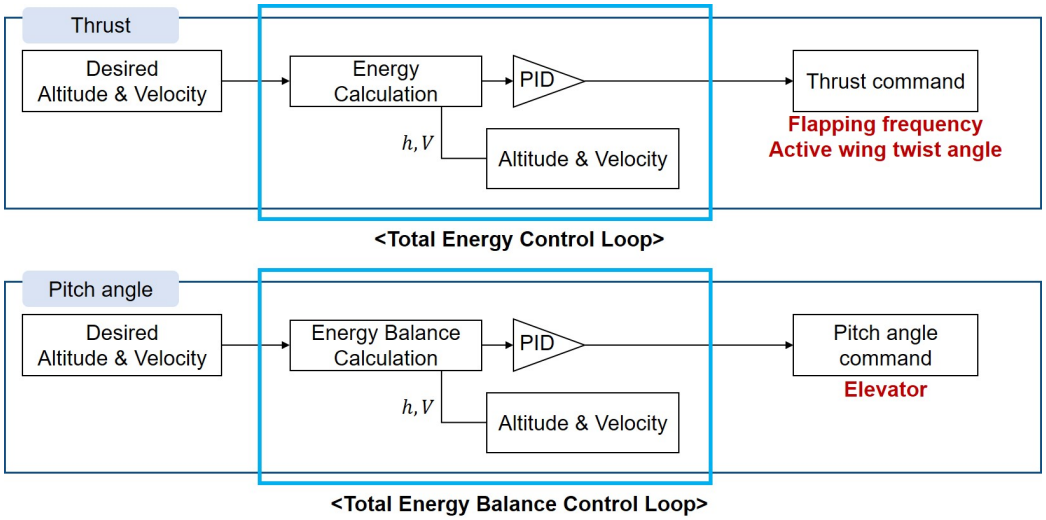


Figure 7: Modified TECS controller

The flapping frequency is controlled when the upstroke state starts, and maintained during that flapping-motion. The active wing twist angle is controlled in real-time. The flapping frequency increases to generate the thrust when the flight speed of the aircraft is lower than the desired flight speed, and the active wing angle decreases to compensate for the loss of the lift caused by the flapping frequency. On the other way, both control inputs operate in reverse of the way above when the flight speed of the aircraft is higher than the desired flight speed.

Stroke-ratio control is not used in the modified TECS controller. It's because both flapping frequency and stroke-ratio are controlled through the electric motor operation. In order to control two elements with only one electric motor, rapid rpm control is required. However, it lacks feasibility when applied to an actual aircraft, so the stroke-ratio is fixed to an optimal value.

4 Controller Performance Analysis

4.1 Simulation configuration

A simulation is configured to analyze the TECS controller design result, and its contents are as follows. The aircraft takes off from the ground at a speed of 6 m/s and climbs to an altitude of 30 m. After reaching the desired altitude, the aircraft cruises maintaining it. Desired flight speed is set to 6 m/s in the entire flight state, and the pitch angle command has the limitation of plus/minus 20 °. There is no disturbance, and the simulation time is set to 300 s.

4.2 Controller performance analysis

Figures 8-9 show the simulation results. The aircraft reaches the desired altitude at about 100 s and cruises maintaining it. The elevator controls the aircraft's pitch angle, and the flapping frequency and wing twist angle control the flight speed according to the command. Flapping frequency decreases as the aircraft reaches the desired altitude, and wing twist angle control occurs actively to reduce flight speed at the beginning of the cruise state. Vibrations due to flapping-motion are shown in all states.

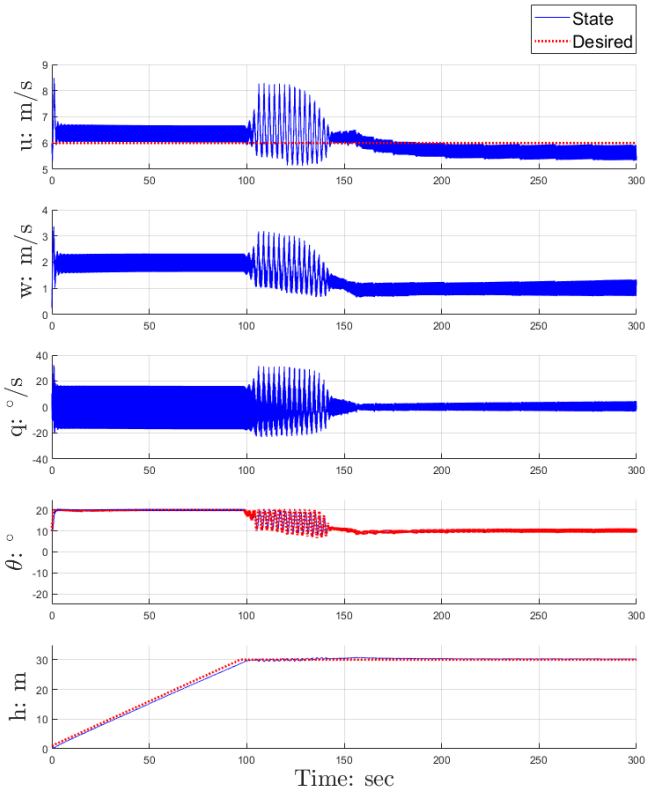


Figure 8: System states

http://www.imavs.org/

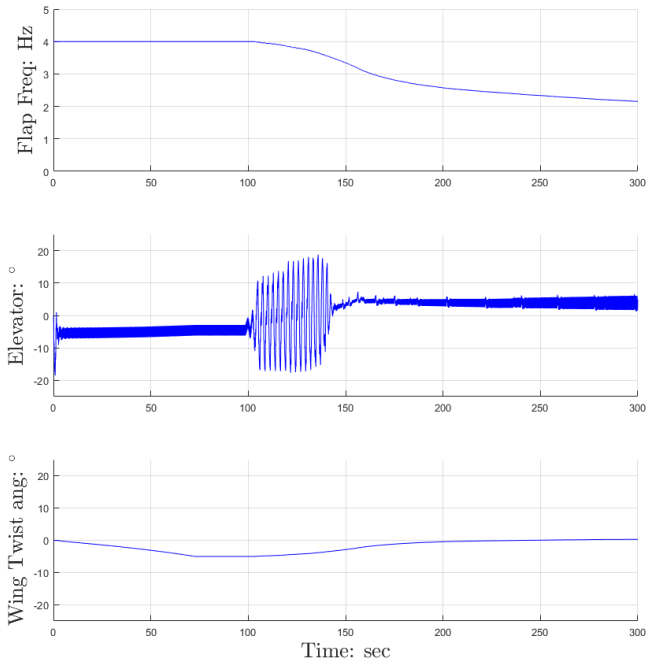


Figure 9: Control inputs

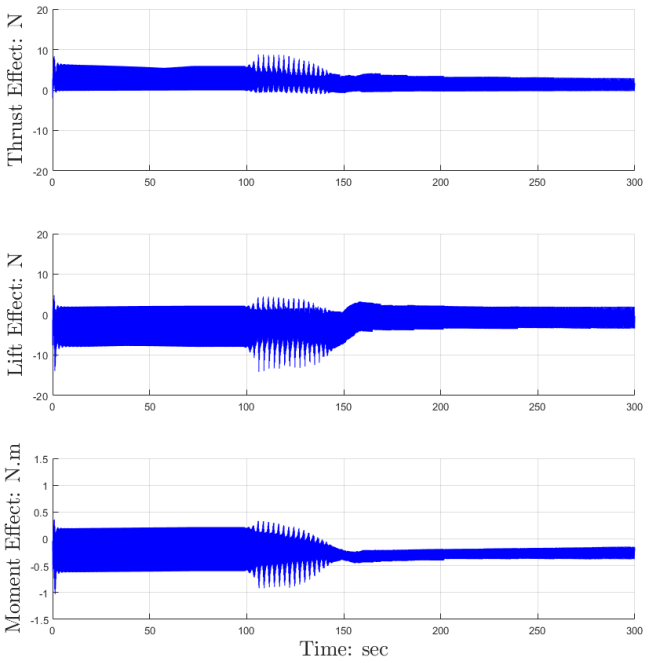


Figure 10: Flapping-motion effects

In the climb flight state, the target altitude is calculated in real-time, and in this state target altitude is higher than the current state. Because the total energy balance control loop controls elevator based on the difference between the kinetic energy and the potential energy, the current flight speed becomes higher than the desired speed. After the aircraft reaches the desired altitude and the cruise flight begins, the TECS controller reduces flight speed, but it has an error with the desired speed. TECS controller's control gain tunning is required to eliminate the error. However, this error is not decreased even with gain tunning for controllability improvement, and an increase in vibrations of system state occurs. So, the error increases contrary to what is intended.

Figure 10 represents the force and moment effects of flapping-motion. The thrust effect follows the trend of change in the flapping frequency, and the wing twist angle has some effects on it. The lift effect has a similar trend above, but with the opposite sign. Through this, it can be confirmed that the flapping frequency effect is greater than the active wing twist angle effect. The elevator affects the pitch moment, and it is confirmed that excessive operation of the elevator causes unstability on the entire force and moment effects through the simulation results for the period of 100 to 150 seconds.

5 Conclusion

In this paper, longitudinal dynamic modeling and controller were designed aimed at a complex articulated ornithopter. The aerodynamic coefficients and the moment of inertia that change according to the flapping-motion were derived, and the equations of motion reflecting the flapping-motion effects were designed. Control inputs for the ornithopter were selected and analyzed based on the analyzed dynamics. Flapping frequency and active wing twist angle control were selected as control inputs, and stroke-ratio was determined as the optimal value considering feasibility. A modified TECS controller was designed and applied taking into account the characteristics of the articulated ornithopter. A simulation for analyzing the designed controller was configured, and its control performances were analyzed with this. As a result, it was confirmed that precise control was difficult with only the TECS controller. Further research for precise control will be continued, and then research on optimal control for the ornithopter will be conducted in the future.

Acknowledgements

This research was supported by Unmanned Vehicles Core Technology Research and Development Program through the National Research Foundation of Korea (NRF) and Unmanned Vehicle Advanced Research Center (UVARC) funded by the Ministry of Science and ICT, the Republic of Korea (2020M3C1C1A01084209).

References

[1] A Lambregts. Vertical flight path and speed control autopilot design using total energy principles. In *Guidance and Control Conference*, page 2239, 1983.

[2] Jong Tai Jang and Sanghyuck Han. Analysis for vtol flight software of px4. In *2018 18th International Conference on Control, Automation and Systems (ICCAS)*, pages 872–875. IEEE, 2018.

http://www.imavs.org/

[3] D Kumar, T Goyal, S Kamle, PM Mohite, and EM Lau. Realisation and testing of novel fully articulated bird-inspired flapping wings for efficient and agile uavs. *The Aeronautical Journal*, 125(1294):2114–2148, 2021.

[4] Wolfgang Send, Markus Fischer, Kristof Jebens, Rainer Mugrauer, Agalya Nagarathinam, and Felix Scharstein. Artificial hinged-wing bird with active torsion and partially linear kinematics. In *Proceeding of 28th Congress of the International Council of the Aeronautical Sciences*. Brisbane, Australia, 2012.

[5] Seongyoung Kim, Myungkang Kim, Seungkeun Kim, and Jinyoung Suk. Design, fabrication, and flight test of articulated ornithopter. In *10th International Micro Air Vehicles Conference, Australia*, pages 1–6, 2018.

[6] Zhao-min Hua, Yu Hou, and Jian-yang Zhu. Aerodynamic analysis of three-section flapping wing by numerical simulation. In *8th International conference on manufacturing science and engineering, advances in engineering research*, volume 164, pages 584–591, 2018.

http://www.imavs.org/

Frequency based strategy for hybrid-powered unmanned aerial vehicle

T. Pavot[1]*, M. Lefebvre[1], R. Kiefer[1], T. Mesbahi[1], E. Laroche[2]
[1]INSA Strasbourg, ICube laboratory UMR7357, 24 Boulevard de la Victoire, Strasbourg, France
[2] Strasbourg University, 3-5 rue de l'université, Strasbourg, France

http://www.imavs.org/

ABSTRACT

Recent developments of different energy sources such as fuel cells (FC), batteries and supercapacitors (SC) show great potential to increase the lifetime and endurance of electric unmanned aerial vehicles (UAV). By knowing the flight cycle, it is possible to optimize the flight time by playing on the advantages and disadvantages of each source. This paper proposes to evaluate the influence of the cutoff frequency to be chosen in order to optimize the lifetime of the hybrid system. To resolve this, a frequency strategy allowing to have a variable cut-off frequency according to the energy stored by the supercapacitors (SC) is developed.

1 INTRODUCTION

From 2017 to 2020, the INSA Strasbourg team developed the Stork drone (Fig.1) as part of the European INTERREG ELCOD project (Endurance Low COst Drone[1]). This drone has been designed to perform long endurance flights for various types of missions, including the measurement of air pollutants. It has a maximum take-off weight of 25 kg (MTOW = 25 kg) and a wingspan of 5 m. This UAV is powered by a hybrid source based on a hydrogen fuel cell associated with a lithium polymer (LiPo) battery. From the tests carried out, the hybridization system of type "all or nothing" proposed by the majority of the manufacturers shows that it is far from being optimal and has limitations.

The objective of this work is to propose a structure and a control of the different hybridization sources in order to optimize the energy management and to improve their lifetime. The Stork UAV and its flight cycle will be used as an typical application to meet the objective of improving energy management within a long endurance UAV by hybridizing three different sources [1]. : a 1000 W PEMFC fuel cell (FC) [2], a LiPo battery and supercapacitors (SC).

This paper will focus on the implementation of a frequency-based power management strategy for hybridization. First, hybridization and sources will be introduced. Then, the strategy with a fixed cutoff frequency will then be discussed with a comparison between simulation and test bench implementation. Finally, the strategy based on a variable cutoff frequency depending on the SC state of charge (SoC) will be compared to the use of a fixed frequency in simulation.



Figure 1: Stork UAV

2 HYBRIDIZATION STRUCTURES

The three sources (FC, SC and batteries) can be classified by their energy density according to their power density. This graph is called the Ragone diagram (Fig.2).

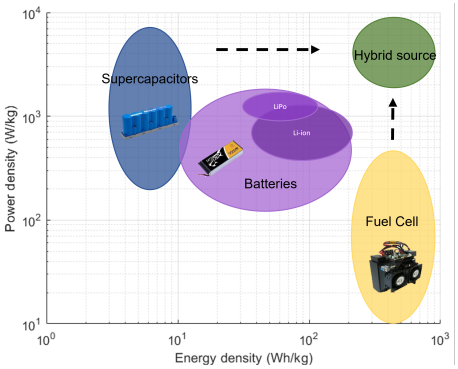


Figure 2: Ragone plot

The interest of creating a hybrid system by associating these three sources, is to allow the system to have a high energy density while ensuring high power density. In other words, the FC is present to ensure endurance while the SCs ensure power variations. The presence of the battery in our case ensures an auxiliary role by making it possible to bring

*Email address(es): thomas.pavot@insa-strasbourg.fr
[1]www.elcod.eu

the surplus of power during the phase of takeoff, the supply of the system to the starting and to bring a safety in the event of breakdown. The hybridization aims to extend the lifetime of the sources while ensuring the maximum autonomy with the constraints of mass and volume inerrant to the drone. The first triple hybridization was performed in 2018. It was a passive hybridization [3].The literature proposes different hybridization architectures depending on the configuration [4].

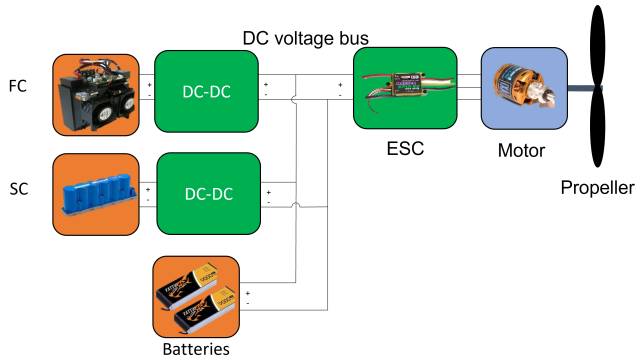


Figure 3: Semi-active architecture

The basic architecture chosen for our system is a so-called semi-active structure (Fig.3). It allows the control of the FC, and the SC by the control of two DC-DC converters. As for the battery, it ensures a stable voltage on the DC bus. This architecture allows the battery to provide additional energy to the system during the entire ascent phase. This surplus of energy cannot be provided by the FC and the SC limited by the converters used.

The control of these sources will be done by the activation of two DC-DC converters able to adapt the voltages and to transfer the powers from a source of energy to another according to the energy needs of the drone. This system uses a minimum of converters thus reducing the number of components and consequently the mass and the price.

3 Sources and DC-DC converters

3.1 Fuel cell (FC)

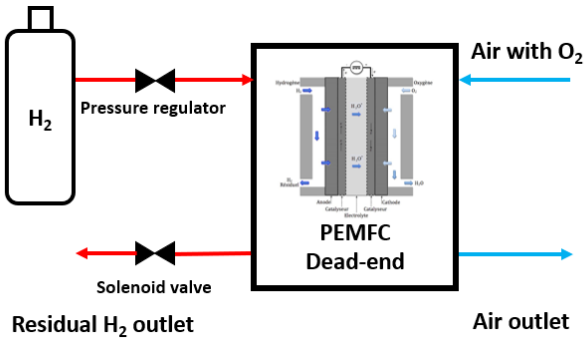


Figure 4: PEMFC dead-end

Table 1: Aerostack PEM fuel cells HES energy system

Specification	Value
Number of cells	50
Max voltage	50V
Voltage at 1000W	33V
Rated power	1000 W
Weight	2182 g
Max $H-2$ flow rate	15 slpm @ 1000 W
$H-2$ pressure	0.55 to 0.77 bar
Peak Efficiency	50%
Humidification	Self-humidified

The FC used is a PEMFC (Proton Exchange Membrane Fuel Cell) of the dead-end type (Fig.4). This type of FC uses as reagent dihydrogen stored at high pressure (300 bars) in an on-board tank which will react with dioxygen present in the air. The dead-end type means that the anodic conduit where the dihydrogen transits is closed [5].

This technology is the lightest and most compact of the different FC technologies. It is suitable for an on-board application in a drone. Its only limit is the flight altitude since the reaction needs a sufficient concentration of oxygen to work. A limit of 3000 m is given by the manufacturer of the FC.

The evolution of the voltage of a PEMFC presents a slow dynamic of a few seconds depending in particular on the humidity of the membrane[6]. The strong variations of current lead to variations of voltage causing a wear of the FC. The literature shows that control with low dynamics decreases the degradation of PEMFCs [7].

The FC used is a 1000W PEMFC aerostack (Table.1) from HES energy system associated to a 6L tank able to contain dihydrogen at 300bar.

3.2 LiPo batteries

LiPo lithium polymer batteries are widely used in electronic devices (smartphones and laptops) and in model making because of their high discharge current and their mass comparable to other types of lithium batteries. However, they are more expensive and have a shorter lifetime compared to lithium-ions which are present in electric vehicles.

The literature clearly shows that current and temperature are the main causes of degradation, whether it is due to time or use. The wear caused will be seen on a decrease of the storage capacity and on the maximum power that can provide the battery [8].

Just like the PEMFC, the LiPo batteries deteriorate with the variations of current and temperature [9, 10].

The batteries used during the flight of the drone are LiPo 9000mAh batteries with a nominal voltage of 14.8V (Table.2). They are associated in series in order to have approximately 32V.

http://www.imavs.org/

Table 2: Tattu LiPo Batteries

Specification	Value
Number of cells	4
Nominal voltage	15.6 V
Weight	809 g
Max current	225 A

3.3 Supercapacitors (SC)

Supercapacitors are electrical dipoles capable of storing a small amount of energy by rapidly charging and discharging. The use of SCs is growing rapidly, they are used in many applications (hybrid vehicles, military warheads, telecommunication, laser technology and photovoltaic panels) [11] to supplement a main power source (often batteries).

The SCs used here are oversized for the application because they are removed off the shelf. It is a block of 6 Maxwell SCs in series with a capacity of 58F (Table.3).

Table 3: Maxwell 16V 58F SC

Specification	Value
Part Number	BMOD0058 E016 B02
Capacity	58F
Max current	12 A continuous & 200 A burst
Max nominal voltage	16 V
Weight	630 g
Max current	225 A

3.4 DC-DC converters

The two converters have an architecture composed of two mosfets in half bridge controlled in reverse associated with a coil and a capacitor. The BuckBoost topology (Fig.5) allows to make these converters unidirectional in voltage and bidirectional in current. The switching frequency used is 20kHz and they are limited to 500 W on the test bench.

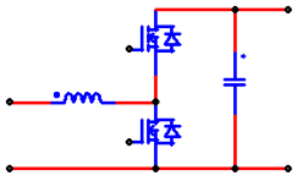


Figure 5: DC-DC converters

4 HYBRIDIZATION STRATEGIES

To operate these converters efficiently, the EMS (Energy Management Strategy) is responsible for ensuring that the energy mix is right for the drone's needs at all times, while maximizing the autonomy and lifetime of the energy sources. The design of an EMS is built by defining a reference flight cycle, objectives (low energy consumption, lifetime ...) and operating criteria (power sources, limits ...). Once this framework is defined, the literature proposes several categories of EMSs [12] for source hybridization :

- Strategies defined by rules (Rule-Based) [13, 14]
- Intelligent strategies based on neural networks or fuzzy logic (Inteligent-Based) [15, 16]
- Optimization strategies (Optimization-Based) [17, 18]
- Other strategies such as frequency cutting [19, 20]

In this paper, the frequency method (Fig.6) has been chosen since it requires low computational power and can easily be implemented in a UAV. This method also allows to respect a sufficiently slow response time in order not to damage the FC by leaving the strong dynamics to the SC. This EMS strategy limits the current variations provided by the FC thus reducing its wear and extending its life.

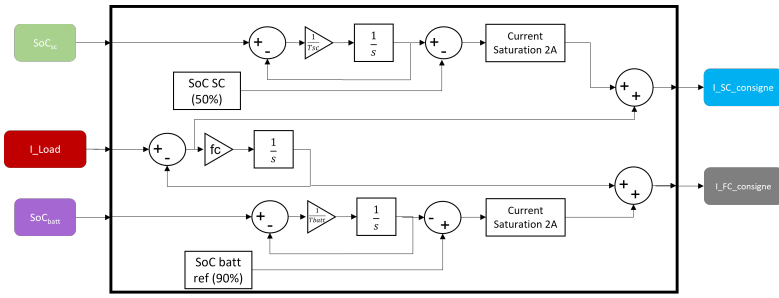


Figure 6: EMS : cut-off frequency

In order to distinguish low frequencies from high frequencies, a first cut-off frequency fc must be determined. The value of $fc = 0.5Hz$ was chosen to allow the fuel cell to respond to the maximum step that the system can undergo. This cut-off frequency also avoids discharging the SC too much during this same step by losing about 10% of its initial state of charge (SoC). This value of cutoff frequency was used as a basis for the continuation by using it on the model and on the test bench. The rest of the article will discuss how to determine the right cutoff frequency for the system.

The second role of the EMS is to maintain a level of SoC for the battery and for the SC. The system will try to maintain the SoC of SC at 50% with a slow feedback loop. This value allows the SC to be able to handle current variations without discharging or overloading itself. In the same way, the battery will be maintained at a SoC of 80%. The battery will be discharged during the phase of takeoff in order to provide the surplus of necessary energy (approximately 1000W). Then, at the time of the cruise flight, the EMS will have for role to recharge then to maintain the SoC of the battery. The value of 80% ensures an autonomy in case of failure of about 30 minutes with the current dimensioning.

http://www.imavs.org/

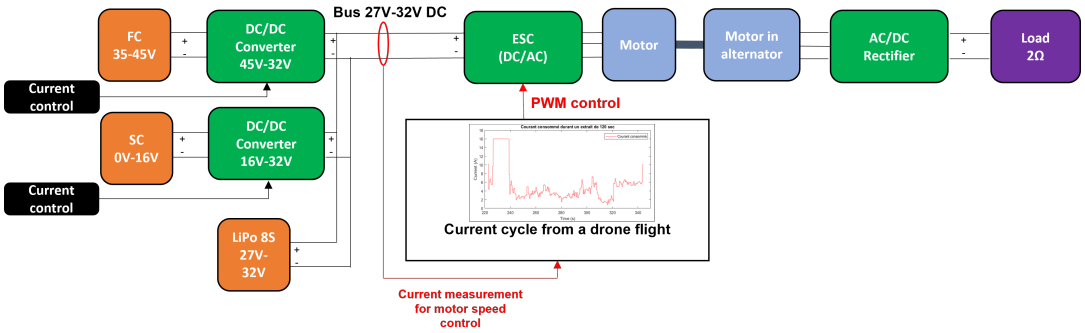


Figure 7: Diagram of the test bench

5 Numerical modeling and experimental validation

To test the EMS, a numerical simulation was performed using the Matlab Simulink software. The modeling of each of the three sources is based on equivalent electrical models. These models easily represent the dynamic behaviors and require a low computation time [21, 22]. Concerning the converters, the modeling is made from an average model not taking into account the chopping caused by the switching of the transistors. These converters are controlled in current with a simple regulation loop using PI (Proportional Integrator) correctors. These models put end to end allowed to have a first representation of the complete functioning of the chosen hybridization architecture.

In order to validate the modeling, a test bench has been realized allowing to perform the same tests. The test bench is represented by its functional diagram on the Fig.7. It is controlled by a Dspace interface (RTI1104) performing the acquisition of voltages and currents and the sending of cyclic control reports to the converters. The load is composed of the ESC (Electronic Speed Controler) and the drone's motor, which are coupled to a second motor/alternator that acts as a brake, dissipating energy through a resistor. The ESC control simulates a current cycle from a real flight of the drone at the nominal operating point of the cruise flight (450 W) to obtain a realistic behavior.

To make the modeling and test bench work, a reference flight cycle that drives the motor must be defined. The ELCOD project has made it possible to carry out flights with the Stork drone allowing to have a realistic current cycle. The selected test is an extract of 120 sec (Fig.8 (a)) of a flight which lasted 30 min including a phase of rise and a phase of flight of cruising. To adapt to the current limit of the bench, the current cycle was divided by 3 and saturated at 16A. These same limitations were applied to the modeling.

The Fig.8 allows to compare the results of the modeling with the experimental test. In Fig.8 (b)(c)(d), the curves show the currents from the simulation compared to those obtained during the experimental test. These are the currents from

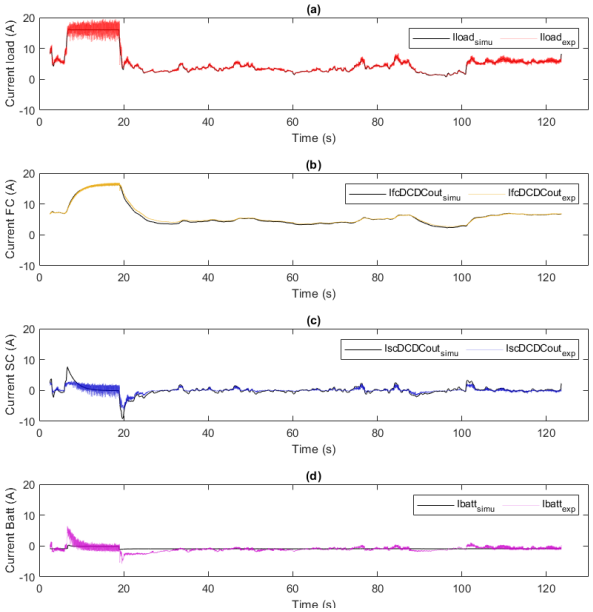


Figure 8: Illustration showing a comparison between the test bench results and the proposed model currents. (a): load current. (b): FC current. (c): SC current. (d): Battery currents.

the FC, SC and batteries (yellow, blue and purple respectively). The blue (IscDCDCout) and yellow (IfcDCDCout) curves represent the output current of each converter. The curves from the simulation are in dark line while the curves from an experimental test are in colored line. The frequency strategy (here with $fc = 0.5Hz$) smoothes the current supplied by the fuel cell. The SC fulfills its role by reacting to fast current demands.

The curves are consistent between the simulation and the experimental. However, during the current draw caused by the load at about 6 sec, we note a difference between the simulation and the experimental test for the battery and the SC. Indeed, in simulation, the battery delivers a current close to 0A, but during the test it compensates the high frequency variations which should be filled by the SC. The blue curve of

http://www.imavs.org/

the test shows that the SC does not manage to follow the set point during the current call. This is for the moment a defect in the operation of the converter. Apart from this fault, the model corresponds to the experimental test. This test allows to validate the operation and now allows to go further in the modeling by testing new variants of the frequency method as EMS.

6 FREQUENCY METHOD: FIXED CUT-OFF FREQUENCY VS VARIABLE CUT-OFF FREQUENCY

The frequency method gives good results by having a cut-off frequency adapted to its system. It is however difficult to determine the best value allowing not to discharge or overload the SC while keeping a sufficiently slow response time not to damage the FC without knowing the flight cycle in advance. The method proposed here allows to vary the cutoff frequency according to the SoC of the SC.

6.1 Theory

The EMS concept, entails delivering the low frequency components of power to the energy source (FC) and the high frequency components to the power source (SCs). A low-pass filter with a variable cut-off frequency f_c is used to achieve this distribution. This last one has an immediate impact on the power restrictions imposed to the FC. The proposed approach in this paper consists on adjusting the cut-off frequency of the filter in real time based on the SoC_{sc} value and reacting fast to load variations. The use of a variable frequency means that there is a high and a low limit to the frequencies used such as $fc_{Hlim} > fc_{Llim}$. These limits are calculated to avoid damaging the SC while maximizing its use. To set these limits, the calibration is done by applying the highest current step that the system can undergo with a fixed cutoff frequency. Having an initial SoC of 50%, the SC has to be able to charge or discharge by 25% to the maximum while undergoing the step. This allows to keep a good efficiency of use of the SC. By respecting this criterion, the minimum cutoff frequency (fc_{Llim}) is 0.125Hz for our system. Conversely, when the SoC of the SC is close to full discharge or overload, it should not undergo a variation of the SoC greater than 5% of its value. This criterion allows in our case to have a high cutoff frequency (fc_{Hlim}) at 1 Hz.

$$fc(t) = (F_{cHlim} - F_{cLlim}) \cdot e^{-5 \cdot SoC(t)} + F_{cLlim} \quad (1)$$

The equation (Eq.1) allowing to link the SoC to the cut-off frequency while taking into account the two previous limits is an exponential equation. This type of equation allows to keep an important dynamics of the system when the SoC evolves around the reference value (50%) while when it moves away enough from this value, the SC becomes much less reactive. The equation (Eq.1) changes if the SC has to charge or discharge. When the SC discharges $SoC(t)$ becomes $1 - SoC(t)$.

6.2 Result and analysis of the simulation

In order to compare the two frequency methods. Three fixed frequencies were chosen. The first is 0.125Hz corresponding to F_{cLlim}. It is the value which makes it possible to completely discharge our SC at the time of the current step undergone at takeoff. This value is problematic because the efficiency of the SC when its SoC drops below 25% is increasingly worse. Indeed to maintain the power at the output of the converter, the current at the output of the SC will increase considerably causing losses by Joule effect. The second value of 1Hz selected corresponds to the F_{cHlim}. This value is too slow and does not allow the SC to be used to its full potential and causes significant variations in the current supplied by the FC. The third and last value is 0.3Hz allowing to have a discharge of the SoC of 25%. This is the fixed cutoff frequency that seems the most optimal on our system for the 120 sec test.

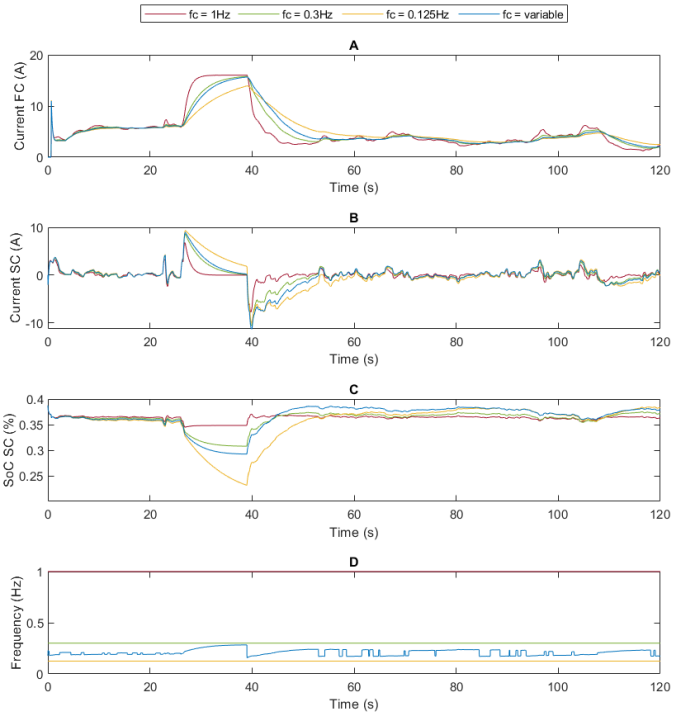


Figure 9: Illustration showing a comparison between different fixed and variable cut-off frequencies, (a): FC current. (b): SC Current. (c): Battery current. (d): SoC of SC. (e): Frequency.

The Fig.9 shows the current curves of each source and the evolution of the SoC of the SC and the cutoff frequency on the 120 sec test. It is interesting to note that the variable frequency allows to have a dynamic close to the fixed frequency of 0.125Hz when the SoC of the SC is close to 0.5 while respecting the discharge during the step of approximately 25% as that of 0.3Hz. These advantages make it possible to better exploit the dynamics of the SC by leaving a slow response

http://www.imavs.org/

time for the FC. The current variations of the FC are therefore lower with a low cut-off frequency allowing an extension of the lifetime. (Fig.9, A). From a safety point of view, the equation allows to avoid at any time to have the SoC of the SC approaching 0 or 1 (Fig.9, B and C) with an approximate setting of the limits. The variable frequency allows the system to maintain a higher dynamic range of the SC without the risk of damaging it with too much charge or discharge.

7 CONCLUSION

In this study, we've first presented the Stork UAV with its power system composed of SC, FC and batteries. Following that, we developed a model of our system and an experimental test bench. The tests conducted allowed us to validate the model with a fixed frequency strategy. We provided an approach to determining the low-pass filter EMS parameters and tuning fc using the SC's SoC. This approach enables the SCs to run with high dynamics while avoiding a SoC_{sc} of 25% or below where efficiency is limited. As a result, this solution should extend the FC's life. In addition to that, this technique allows to better protect the SCs from overcharging/discharging. The future prospects of this work include the search for reliable indicators to prove that the proposed solution brings benefits for the autonomy and the lifetime of the UAV. The comparison with new EMS algorithms such as robust control or optimal control will be studied in order to provide a comparison with the frequency method developed here.

ACKNOWLEDGEMENTS

The authors thank ICUBE laboratory from Strasbourg University and INSA Strasbourg for funding this study.

REFERENCES

[1] M. Boukoberine, Z. Zhou, and M. Benbouzid. A critical review on unmanned aerial vehicles power supply and energy management: Solutions, strategies, and prospects, dec 2019.

[2] Z. F. Pan, L. An, and C. Y. Wen. Recent advances in fuel cells based propulsion systems for unmanned aerial vehicles, apr 2019.

[3] A. Gong and D. Verstraete. Flight Test of a Fuel-Cell / Battery / Supercapacitor Triple Hybrid. *31st Congress of the International Council of the Aeronautical Sciences*, (September):1–10, 2018.

[4] B. Wang, D. Zhao, W. Li, Z. Wang, Y. Huang, Y. You, and S. Becker. Current technologies and challenges of applying fuel cell hybrid propulsion systems in unmanned aerial vehicles. *Progress in Aerospace Sciences*, 116(May), 2020.

[5] S. Abbou. *Phénomènes locaux instationnaires dans les piles à combustible à membrane (PEMFC) fonctionnant en mode bouché (dead-end)*. PhD thesis, 2015.

[6] Ph Moçotéguy, F. Druart, Y. Bultel, S. Besse, and A. Rakotondrainibe. Monodimensional modeling and experimental study of the dynamic behavior of proton exchange membrane fuel cell stack operating in dead-end mode. *Journal of Power Sources*, 167(2):349–357, may 2007.

[7] E. Pahon, S. Jemei, N. Steiner, and D. Hissel. Effect of load cycling on the performance of fuel cell stacks. *2019 IEEE Vehicle Power and Propulsion Conference, VPPC 2019 - Proceedings*, 2019.

[8] E. Iglesias. Étude du vieillissement des batteries lithium-ion dans les applications "véhicule électrique" : Combinaison des effets de vieillissement calendaire et de cyclage. Technical report, 2017.

[9] R. Xiong, Y. Pan, W. Shen, H. Li, and F. Sun. Lithium-ion battery aging mechanisms and diagnosis method for automotive applications: Recent advances and perspectives, oct 2020.

[10] C. Vidal, O. Gross, R. Gu, P. Kollmeyer, and A. Emadi. XEV Li-Ion Battery Low-Temperature Effects-Review. *IEEE Transactions on Vehicular Technology*, 68(5):4560–4572, may 2019.

[11] W. Raza, F. Ali, N. Raza, Y. Luo, K. Kim, J. Yang, S. Kumar, A. Mehmood, and E. Kwon. Recent advancements in supercapacitor technology, oct 2018.

[12] T. Lei, Z. Yang, Z. Lin, and X. Zhang. State of art on energy management strategy for hybrid-powered unmanned aerial vehicle, jun 2019.

[13] B. Lee, S. Kwon, P. Park, and K. Kim. Active power management system for an unmanned aerial vehicle powered by solar cells, a fuel cell, and batteries. *IEEE Transactions on Aerospace and Electronic Systems*, 50(4):3167–3177, 2014.

[14] Y. Han, W. Chen, and Q. Li. Energy Management Strategy Based on Multiple Operating States for a Photovoltaic/Fuel Cell/Energy Storage DC Microgrid. 2017.

[15] X. Zhang, L. Liu, Y. Dai, and T. Lu. Experimental investigation on the online fuzzy energy management of hybrid fuel cell/battery power system for UAVs. *International Journal of Hydrogen Energy*, 43(21):10094–10103, 2018.

[16] H. Khayyam and A. Bab-Hadiashar. Adaptive intelligent energy management system of plug-in hybrid electric vehicle. *Energy*, 69:319–335, 2014.

[17] V. Larsson, L. Johannesson, and B. Egardt. Analytic solutions to the dynamic programming subproblem in hybrid vehicle energy management. *IEEE Transactions on Vehicular Technology*, 64(4):1458–1467, 2015.

[18] P. García, J. P. Torreglosa, L. M. Fernández, and F. Jurado. Viability study of a FC-battery-SC tramway controlled by equivalent consumption minimization strategy. *International Journal of Hydrogen Energy*, 37(11):9368–9382, 2012.

[19] H. Xiaoliang, T. Hiramatsu, and H. Yoichi. Energy management strategy based on frequency-varying filter for the battery supercapacitor hybrid system of Electric Vehicles. *World Electric Vehicle Journal*, 6(3):623–628, 2013.

[20] J. Snoussi, S. Ben Elghali, M. Benbouzid, and M. Mimouni. Auto-adaptive filtering-based energy management strategy for fuel cell hybrid electric vehicles. *Energies*, 11(8):1–20, 2018.

[21] F. Belhaj, H. El Fadil, Z. El Idrissi, A. Intidam, M. Koundi, and F. Giri. New equivalent electrical model of a fuel cell and comparative study of several existing models with experimental data from the PEMFC nexa 1200 W. *Micromachines*, 12(9), 2021.

[22] Y. Ghoulam, T. Paul, T. Mesbahi, and S. Durand. Modeling, Identification and Simulation of Hybrid Battery/Supercapacitor Storage System Used in Vehicular Applications. *6th International Conference on Electric Vehicular Technology 2019*, (November):156–162, 2019.

http://www.imavs.org/

Bibliography

[1] Marco A. Martinez-Ramirez and Hugo Rodríguez-Cortés. Lateral guidance and control for a fixed-wing aircraft. In G. de Croon and C. De Wagter, editors, *13th International Micro Air Vehicle Conference*, pages 17–24, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-1. 13

[2] Zeynep Bilgin, Murat Bronz, and Ilkay Yavrucuk. Panel method based guidance for fixed wing micro aerial vehicles. In G. de Croon and C. De Wagter, editors, *13th International Micro Air Vehicle Conference*, pages 25–31, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-2. 13

[3] Taegyun Kim, Jingu Kim, Ilwon Seo, and Seungkeun Kim. Reconnaissance target grouping and assignment considering fault severity for uav swarm. In G. de Croon and C. De Wagter, editors, *13th International Micro Air Vehicle Conference*, pages 32–38, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-3. 13

[4] Gautier Hattenberger, Murat Bronz, and Jean-Philippe Condomines. Evaluation of drag coefficient for a quadrotor model. In G. de Croon and C. De Wagter, editors, *13th International Micro Air Vehicle Conference*, pages 39–46, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-4. 13

[5] Stavrow Bahnam, Christophe de Wagter, and Guido de Croon. Improving the computational efficiency of rovio. In G. de Croon and C. De Wagter, editors, *13th International Micro Air Vehicle Conference*, pages 47–52, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-5. 13

[6] Carlos Alexander Quero, Daniel Durini, José de Jesús Magdaleno, Jose Martinez-Carranza, and Ruben Ramos-Garcia. Fast nir-single-pixel-imaging enhancement under scattering environment. In G. de Croon and C. De Wagter, editors, *13th International Micro Air Vehicle Conference*, pages 53–58, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-6. 13

[7] Elia Cereda, Daniele Palossi, and Alessandro Giusti. Handling pitch variations for visual perception in mavs: Synthetic augmentation and state fusion. In G. de Croon and C. De Wagter, editors, *13th International Micro Air Vehicle Conference*, pages 59–65, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-7. 13

[8] Lucia Bergantin, Charles Coquet, Amaury Negre, Thibaut Raharijaona, Nicolas Marchand, and Franck Ruffier. Using trajectory oscillation timing improves in-flight odometry based solely on optic flows. In G. de Croon and C. De Wagter, editors, *13th International Micro Air Vehicle Conference*, pages 66–74, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-8. 13

[9] Simon Wilshin, Stephen Amos, and Richard J Bomphrey. Seeing with sound – surface detection and avoidance by sensing self-generated noise. In G. de Croon and C. De Wagter, editors, *13th International Micro Air Vehicle Conference*, pages 75–83, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-9. 13

[10] Ze Wang, Jingang Qu, and Pascal Morin. Deep learning-based flight speed estimation using thermal anemometers. In G. de Croon and C. De Wagter, editors, *13th International Micro Air Vehicle Conference*, pages 84–91, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-10. 13

[11] Miranda-Varela Mariana-Edith, Leticia Oyuki Rojas-Perez, and Jose Martinez-Carranza. Semantic segmentation of rgb aerial images using cnn and superregions. In G. de Croon and C. De Wagter, editors, *13th International Micro Air Vehicle Conference*, pages 92–102, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-11. 13

[12] Aldrich Alfredo Cabrera-Ponce, Manuel Isidro Martin-Ortiz, and Jose Martinez-Carranza. Multi-model continual learning for camera localisation from aerial images. In G. de Croon and C. De Wagter, editors, *13th International Micro Air Vehicle Conference*, pages 103–109, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-12. 13

http://www.imavs.org/

[13] L. Oyuki Rojas-Perez and Jose Martinez-Carranza. Leveraging a neural pilot via automatic gain tuning using gate detection for autonomous drone racing. In G. de Croon and C. De Wagter, editors, *13^{th} International Micro Air Vehicle Conference*, pages 110–118, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-13. 13

[14] Dennis van Wijngaarden and Bart D.W. Remes. Indi control for the oblique wing-quad plane drone. In G. de Croon and C. De Wagter, editors, *13^{th} International Micro Air Vehicle Conference*, pages 119–126, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-14. 13

[15] Julian Müller and Dieter Moormann. Miniaturization and control of an unmanned tiltwing aircraft. In G. de Croon and C. De Wagter, editors, *13^{th} International Micro Air Vehicle Conference*, pages 127–135, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-15. 13

[16] Daniel Schatten. Attitude control for a tiltwing aircraft under tail actuator failures. In G. de Croon and C. De Wagter, editors, *13^{th} International Micro Air Vehicle Conference*, pages 136–143, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-16. 13

[17] Hoijo Jeaong, Jinyoung Suk, and Seungkeun Kim. Robust flight control using hybrid incremental nonlinear dynamic inversion control for tiltrotor uav. In G. de Croon and C. De Wagter, editors, *13^{th} International Micro Air Vehicle Conference*, pages 144–151, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-17. 13

[18] Martin Lefebvre, Thomas Pavot, Renaud Kiefer, Marc Vedrines, and Nadia Bahlouli. Recyclable bio-based composite flax/elium for uav applications. In G. de Croon and C. De Wagter, editors, *13^{th} International Micro Air Vehicle Conference*, pages 152–158, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-18. 13

[19] Luiz Fernando Tiberio Fernandez, Murat Bronz, Nathalie Bartoli, and Thierry Lefebvre. Development of a mission-tailored tail-sitter mav. In G. de Croon and C. De Wagter, editors, *13^{th} International Micro Air Vehicle Conference*, pages 159–168, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-19. 13

[20] Beatriz Bernabe-Lorranca and Hugo Rodríguez-Cortés. C wing geometric redesign based on aircraft flying qualities. In G. de Croon and C. De Wagter, editors, *13^{th} International Micro Air Vehicle Conference*, pages 169–174, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-20. 13

[21] Gautier Hattenberger, Fabien Bonneval, Matheus Ladeira, Emmanuel Grolleau, and Yassine Ouhammou. Micro-drone autopilot architecture for efficient static scheduling. In G. de Croon and C. De Wagter, editors, *13^{th} International Micro Air Vehicle Conference*, pages 175–182, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-21. 13

[22] Mohammad Lashgari and Abolghasem Naghash. Hover controller design and implementation for a dragonfly-like flapping wing. In G. de Croon and C. De Wagter, editors, *13^{th} International Micro Air Vehicle Conference*, pages 183–192, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-22. 13

[23] Shawn Schröter, Ewoud J.J. Smeur, and Bart D.W. Remes. Design and joint control of a conjoined biplane and quadrotor. In G. de Croon and C. De Wagter, editors, *13^{th} International Micro Air Vehicle Conference*, pages 193–200, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-23. 13

[24] Woongtaek Oh, Inrae Kim, Seungkeun Kim, and Jinyoung Suk. Total energy control system design for longitudinal dynamics of complex articulated ornithopter. In G. de Croon and C. De Wagter, editors, *13^{th} International Micro Air Vehicle Conference*, pages 201–206, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-24. 13

[25] Thomas Pavot, Martin Lefebvre, Renaud Kiefer, Tedjanie Mesbahi, and Edouard Laroche. Frequency based strategy for hybrid-powered unmanned aerial vehicle. In G. de Croon and C. De Wagter, editors, *13^{th} International Micro Air Vehicle Conference*, pages 207–212, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-25. 13

http://www.imavs.org/