



IMAV 2023

14th annual International Micro Air Vehicle Conference and
Competition

September 11-15, 2023

Aachen, Germany

www.imavs.org



Contents

IMAV2023	1
Preface	3
International IMAV Committee	4
Program Committee	4
Organizing Committee	4
Sponsors	5
Call for Papers	6
Sessions	7
List Of Papers	8
Best Paper Awards	8
Authors	9
Papers	11
High-speed camera measurement of insect-like flapping wing deformation using a speckle pattern wing...	12
Design and Path Following Flight Test of a Flapping-Wing Air Vehicle	20
Integrated Simulation Program for Initial Design of Bird-Inspired Ornithopter	26
3D Human Pose reconstruction Single-pixel imaging	33
Hierarchical Continual Learning for Single Image Aerial Localisation	40
Determining the moment of inertia tensor of a UAV using a motion capture system	49
Why do I need to speak to my drone?	57
A Lightweight Learning-based Visual-Inertial Odometry	65
Frustumbug: a 3D mapless stereo-vision-based bug algorithm for Micro Air Vehicles	73
Improved Moving Horizon Estimation for Ultra-Wideband Localization on Small Drones	86
Efficient Model-Aided Visual-Inertial Ego-Motion Estimation for Multirotor MAVs	93
Analysis of admittance control for a fully-actuated, tactile UAV	101
Autopilot framework with INDI RPM control, real-time actuator feedback, and stability control on...	109
Cascaded Indoor Flight Controller Design for a Miniaturized Tiltwing Aircraft	117
Performance Comparisons on Fully-Actuated Multirotors with Different Actuation Modes	127
Distributed Nonlinear Model Predictive Control for a Quadrotor UAV	135
Adaptive frequency control strategy for the propulsion management of a long-endurance drone using...	143
Model Predictive Control on a Fully-actuated Octocopter for Wind Disturbance Rejection	149
Assessing the Agility of a Variable-Tilt UAV	157
Analysis of the Transient Response of UAV Rotors	165
Multi-Vehicle Simulation Framework for Heterogeneous Unconventional MAVs	171
Experimentally Validated Simulator of Flight Dynamics: for a Magnus Effect-based Quadcopter System	179
Numerical Investigation on Flow Control of Leading-Edge Reinforced Flexible Membrane Wing	187
Measuring Disturbance Response of a Fully-Actuated UAV using Surging Flow Wind Tunnel Experiments	193
Multi Method Local Path Planning Considering Flight Dynamics Constraints for Collision Avoidance	200
Autonomous Landing, Obstacle Avoidance and Block Recovery for a Quadrotor Drone	208
Automating Fixed Wing Forced Landings with Offline Reinforcement Learning	216
Trajectory Optimization for Fully Actuated Hexacopters: Enhancing Maneuverability and Applications	224
Rigid Airborne docking between a fixed-wing UAV and an over-actuated multicopter	231
Task Allocation for Heterogeneous Unmanned Aerial Vehicles	240
Time-Optimal Trajectory Generation with Input Boundaries and Dynamic Waypoints Constraints for A...	247
MTSO: Multi-Target Search Optimisation based on Probability Map	255

Preface

On behalf of the Local Organizing Committee, it is our pleasure to present the proceedings of the 14th annual International Micro Air Vehicle Conference and Competition, which was held in Aachen, Germany from September 11-15, 2023. For the second time, the IMAV was organized by the Institut of Flight System Dynamics of the RWTH Aachen University in Germany.

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

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

These proceedings contain thirty-two peer-reviewed scientific papers by ninety-one authors organized in eight sessions presented at the IMAV in 2023. 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. Six 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, six 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 fourteen members of the International Committee from ten institutes who guide IMAV overall, and the twenty reviewers from twelve institutes, who played an important role in assuring the quality of all the papers. In addition to the presentation of the scientific papers, keynote talks were delivered by experts in the field whom we want to thank for their valuable contributions. We also want to thank all members of the Local Organizing Committee for their invaluable support in the organization of this IMAV-2023.

Aachen, Germany. September 2023

D. Moormann
[RWTH, Aachen](#)

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
 Thipyopas Chinnapat [KU](#), Bangkok, Thailand

Program Committee

Antoine Drouin [ENAC](#), Toulouse, France
 Bart Remes [TUDelft](#), Delft, the Netherlands
 Ben M. Chen [CUHK](#), Hong-Kong, China
 Christophe De Wagter [TUDelft](#), Delft, the Netherlands
 Dieter Moormann [RWTH](#), Aachen, Germany
 Gautier Hattenberger [ENAC](#), Toulouse, France
 Guido de Croon [TUDelft](#), Delft, the Netherlands
 Hector J. Garcia de Marina Peinado . . [UCM](#), Madrid, Spain
 Jean-Marc Moschetta [ISAE](#), Toulouse, France
 José Martinez-Carranza [INAOE](#), Puebla, Mexico

Leandro Lustosa [ISAE-SUPAERO](#), Toulouse, France
 Mohamed Abdulghani [RMIT](#), Melbourne, Australia
 Murat Bronz [ENAC](#), Toulouse, France
 Norbert Siepenkötter [RWTH](#), Aachen, Germany
 Pascual Campoy [UPM](#), Madrid, Spain
 Philipp Hartmann [FH](#), Aachen, Germany
 Sebastian Seitz [RWTH](#), Aachen, Germany
 Simon Watkins [RMIT](#), Melbourne, Australia
 Thipyopas Chinnapat [KU](#), Bangkok, Thailand
 Tobias Ostermann [RWTH](#), Aachen, Germany

Organizing Committee

Dieter Moormann, General Chair
 Sebastian Seitz Chief Jury Outdoor
 Jan Tilmans Chief Jury Indoor
 Daniel Schatten Program Chair
 Dominik Duda Organizing Teamleader
 Eva König Organizing Teamleader
 Julian Müller Organizing Teamleader
 Phillipp Müller Organizing Teamleader
 Tobias Ostermann Staff
 Norbert Siepenkötter Staff
 Tobias Islam Staff
 Lukas Bidus Staff

Lennart Danielmeier Staff
 Hendrik Fuest Staff
 Max Hartmann Staff
 Jan-Phillip Herzog Staff
 Sven Jackisch Staff
 Florian Knaak Staff
 Johannes Krimphove Staff
 Woldemar Neustätter Staff
 Julian Növer Staff
 Patrick Osterloh Staff
 Anton Schwager Staff
 Ann-Kristin Sturm Staff

Sponsors



Call for Papers



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

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

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

Call for papers

We are now inviting authors to submit papers for presentation at the conference. We welcome contributions to all aspects of MAVs, including, but not limited to:

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

Submission Guidelines

During the conference, scientific and technical papers can be presented. Authors will have an allotted time slot of 20 minutes to present their work. Technical and scientific papers can be submitted as draft papers. Full papers should be submitted until September 1, 2023. Papers should have a minimum of 6 pages and a maximum of 8 pages.

All papers must be original and not simultaneously submitted to another journal or conference. File sizes should be minimized (max 5MB) while graphical quality should be maximized by using vector graphics whenever possible.

Venue

- Location: Aachen, Germany
- Event dates: September 11 -15, 2023. Web site: <http://www.imavs.org/2023>

Conference Oral Presentations

Flapping Wings

High-speed camera measurement of insect-like flapping wing deformation using a speckle pattern wing...	p.12
Design and Path Following Flight Test of a Flapping-Wing Air Vehicle.....	p.20
Integrated Simulation Program for Initial Design of Bird-Inspired Ornithopter	p.26

Sensing and perception 1

3D Human Pose reconstruction Single-pixel imaging	p.33
Hierarchical Continual Learning for Single Image Aerial Localisation.....	p.40
Determining the moment of inertia tensor of a UAV using a motion capture system	p.49
Why do I need to speak to my drone?	p.57

Sensing and perception 2

A Lightweight Learning-based Visual-Inertial Odometry.....	p.65
Frustumbug: a 3D mapless stereo-vision-based bug algorithm for Micro Air Vehicles.....	p.73
Improved Moving Horizon Estimation for Ultra-Wideband Localization on Small Drones	p.86
Efficient Model-Aided Visual-Inertial Ego-Motion Estimation for Multirotor MAVs	p.93

New control methods 1

Analysis of admittance control for a fully-actuated, tactile UAV	p.101
Autopilot framework with INDI RPM control, real-time actuator feedback, and stability control on.....	p.109
Cascaded Indoor Flight Controller Design for a Miniaturized Tiltwing Aircraft	p.117
Performance Comparisons on Fully-Actuated Multirotors with Different Actuation Modes	p.127

New control methods 2

Distributed Nonlinear Model Predictive Control for a Quadrotor UAV.....	p.135
Adaptive frequency control strategy for the propulsion management of a long-endurance drone using.....	p.143
Model Predictive Control on a Fully-actuated Octocopter for Wind Disturbance Rejection	p.149

Design of novel MAV types

Assessing the Agility of a Variable-Tilt UAV	p.157
Analysis of the Transient Response of UAV Rotors.....	p.165
Multi-Vehicle Simulation Framework for Heterogeneous Unconventional MAVs	p.171
Experimentally Validated Simulator of Flight Dynamics: for a Magnus Effect-based Quadcopter System	p.179
Numerical Investigation on Flow Control of Leading-Edge Reinforced Flexible Membrane Wing	p.187
Measuring Disturbance Response of a Fully-Actuated UAV using Surging Flow Wind Tunnel Experiments	p.193

Guidance and path planning

Multi Method Local Path Planning Considering Flight Dynamics Constraints for Collision Avoidance.....	p.200
Autonomous Landing, Obstacle Avoidance and Block Recovery for a Quadrotor Drone	p.208
Automating Fixed Wing Forced Landings with Offline Reinforcement Learning	p.216
Trajectory Optimization for Fully Actuated Hexacopters: Enhancing Maneuverability and Applications	p.224

Multi-UAV systems and swarm intelligence

Rigid Airborne docking between a fixed-wing UAV and an over-actuated multicopter.....	p.231
Task Allocation for Heterogeneous Unmanned Aerial Vehicles	p.240
Time-Optimal Trajectory Generation with Input Boundaries and Dynamic Waypoints Constraints for A... ..	p.247
MTSO: Multi-Target Search Optimisation based on Probability Map	p.255

Papers

[1] High-speed camera measurement of insect-like flapping wing deformation using a speckle pattern wing...	p.12
[2] Design and Path Following Flight Test of a Flapping-Wing Air Vehicle	p.20
[3] Integrated Simulation Program for Initial Design of Bird-Inspired Ornithopter	p.26
[4] 3D Human Pose reconstruction Single-pixel imaging.	p.33
[5] Hierarchical Continual Learning for Single Image Aerial Localisation	p.40
[6] Determining the moment of inertia tensor of a UAV using a motion capture system.....	p.49
[7] Why do I need to speak to my drone?.....	p.57
[8] A Lightweight Learning-based Visual-Inertial Odometry	p.65
[9] Frustumbug: a 3D mapless stereo-vision-based bug algorithm for Micro Air Vehicles.....	p.73
[10] Improved Moving Horizon Estimation for Ultra-Wideband Localization on Small Drones.....	p.86
[11] Efficient Model-Aided Visual-Inertial Ego-Motion Estimation for Multirotor MAVs.....	p.93
[12] Analysis of admittance control for a fully-actuated, tactile UAV	p.101
[13] Autopilot framework with INDI RPM control, real-time actuator feedback, and stability control on... ..	p.109
[14] Cascaded Indoor Flight Controller Design for a Miniaturized Tiltwing Aircraft	p.117
[15] Performance Comparisons on Fully-Actuated Multirotors with Different Actuation Modes.....	p.127
[16] Distributed Nonlinear Model Predictive Control for a Quadrotor UAV	p.135
[17] Adaptive frequency control strategy for the propulsion management of a long-endurance drone using... ..	p.143
[18] Model Predictive Control on a Fully-actuated Octocopter for Wind Disturbance Rejection	p.149
[19] Assessing the Agility of a Variable-Tilt UAV	p.157
[20] Analysis of the Transient Response of UAV Rotors	p.165
[21] Multi-Vehicle Simulation Framework for Heterogeneous Unconventional MAVs.....	p.171
[22] Experimentally Validated Simulator of Flight Dynamics: for a Magnus Effect-based Quadcopter System.....	p.179
[23] Numerical Investigation on Flow Control of Leading-Edge Reinforced Flexible Membrane Wing	p.187
[24] Measuring Disturbance Response of a Fully-Actuated UAV using Surging Flow Wind Tunnel Experiments.....	p.193
[25] Multi Method Local Path Planning Considering Flight Dynamics Constraints for Collision Avoidance	p.200
[26] Autonomous Landing, Obstacle Avoidance and Block Recovery for a Quadrotor Drone	p.208
[27] Automating Fixed Wing Forced Landings with Offline Reinforcement Learning	p.216
[28] Trajectory Optimization for Fully Actuated Hexacopters: Enhancing Maneuverability and Applications.....	p.224
[29] Rigid Airborne docking between a fixed-wing UAV and an over-actuated multicopter	p.231
[30] Task Allocation for Heterogeneous Unmanned Aerial Vehicles.....	p.240
[31] Time-Optimal Trajectory Generation with Input Boundaries and Dynamic Waypoints Constraints for A... ..	p.247
[32] MTSO: Multi-Target Search Optimisation based on Probability Map	p.255

Best Paper Awards and Special issues

IMAV 2023 awarded the best paper award to a relevant, well-written, and well-presented publication. It also selected a shortlist of 6 papers to be published in a Special Issue of the International Journal of Micro Air Vehicles (Sage). The papers are [1] [10] [17] [18] [19] [28]

Best Conference Paper

High-speed camera measurement of insect-like flapping wing deformation using a speckle pattern wing membrane.
Thomas Roelandt, Thijs Willems, Frank Naets and Dirk Vandepitte. [1] on page 12.

Authors

A

Aggarwal, Nilesh p.255
 Al-Zubaidi, Salim p.49, p.101, p.157, p.165
 Anunay, Anunay p.255
 Azaki, Zakeye p.179

B

Bartoli, Nathalie p.171
 Bayon, Marcos Linares p.208
 Bronz, Murat p.171, p.224

C

Cabrera-Ponce, Aldrich Alfredo p.40, p.57
 Chen, Junyi p.193
 Croon, Guido C.H.E. de p.65, p.73, p.86, p.93

D

Danielmeier, Lennart p.200
 Dijk, Tom van p.73
 Duda, Dominik Felix p.117
 Dumon, Jonathan p.179
 Durini, Daniel p.33

E

East, Sebastian p.216
 Edwards, Matthew p.157

F

Fei, Matteo p.208
 Fernandez, Luiz Fernando Tiberio p.171

H

Hably, Ahmad p.179
 Hachem, Mohamad p.224
 Han, Jae-Hung p.20, p.26
 Hartmann, Max p.200
 Hernandez-Farias, Delia Irazu p.57
 Horst, Erik Van Der p.109
 Hove, Jonathas Laffita van den p.231

J

Jain, Vayam p.255

K

Kay, Nicholas p.149
 Kiefer, Renaud p.143
 Knaak, Florian p.240

L

Laroche, Edouard p.143
 Lee, Eun-Hyuck p.26
 Lee, Sang-Gil p.20, p.26
 Lee, Wonhee p.208
 Lefebvre, Thierry p.171
 Li, Shuo p.247
 Liu, Iris p.101
 Liu, Zhen p.187

M

Malhotra, Anup Kumar p.255
 Mancinelli, Alessandro p.109
 Martin-Ortiz, Manuel Isidro p.40
 Martinez-Carranza, Jose p.33, p.40, p.57
 Meester, Ruben p.73
 Mei, Jiahao p.247
 Mesbahi, Tedjani p.143
 Meslem, Nacim p.179
 Miquel, Thierry p.224
 Mller, Julian p.117
 Moormann, Dieter p.117, p.200, p.240
 Mubdir, Bilal p.135
 Müller, Philipp p.240

N

Naets, Thijs Willems, Frank p.12
 Negre, Amaury p.179

O

Offermann, Alexis p.179

P

Pavot, Thomas p.143
 Pfeiffer, Sven p.86

Prempain, Emmanuel p.135
 Probine, Caleb p.127, p.149

Q

Quero, Carlos Osorio p.33
 Quessy, Alexander p.216

R

Radu, Andrei Muresan p.208
 Ramos-Garcia, Ruben p.33
 Rangel-Magdaleno, Jose p.33
 Remes, Bart D.W. p.109, p.231
 Ribeiro, Leandro Lustosa p.208
 Richardson, Thomas p.216
 Roelandt, Thomas p.12
 Rojas-Perez, Leticia Oyuki p.57
 Roos, Clément p.224

S

Schatten, Daniel p.117
 Schwager, Anton Sebastian p.240
 Seitz, Sebastian p.200
 Shen, Yuyang p.247
 Singh, Sarthak p.255
 Smeur, Ewoud J.J. p.109, p.231
 Stol, Karl ... p.49, p.101, p.127, p.149, p.157, p.165, p.193
 Susbielle, Pierre p.179
 Swaraj, Suvarna p.101

T

Taylor, Joshua p.157

V

Vandepitte, Dirk p.12
 Virmani, Aman p.255
 Voget, Nicolai p.200

W

Wagter, Christophe De p.73, p.86

X

Xie, Wenhao p.187
 Xu, Jinming p.247
 Xu, Yingfu p.65, p.93

Y

Yang, David p.49, p.149
 Yang, Hyeon-Ho p.20, p.26
 You, Hwankyun p.20

Z

Zhao, Fangguo p.247
 Zhou, Jin p.247

Release

Released on September 12th 2022.

V1.2, Rev.16824, 2026-10-18 13:08
Created by: D. Schatten and C. De Wagter

High-Speed Camera Measurement of Insect-like Flapping Wing Deformation Using a Speckle Pattern Wing Membrane

Thomas Roelandt^{1,*}, Thijs Willems^{1,2}, Frank Naets^{1,2} and Dirk Vandepitte¹

¹KU Leuven, Celestijnenlaan 300 box 2420, 3001 Leuven, Belgium

² Flanders Make@KU Leuven, Leuven, Belgium

ABSTRACT

Wing kinematics and wing deformation have a strong effect on the lift production and aerodynamic efficiency of bio-inspired flapping wing aerial vehicles. The goal of this study is to automatically track and classify wing kinematics components and wing deformation components of an insect-like flapping wing. Insight into wing motion and deformation characteristics may allow for improved performance of flapping wing drones. This work uses a stereoscopic high-speed camera setup and feature detection algorithm to measure the kinematics and deformation components of an insect-like flapping wing system with passive wing inclination. The wing membrane is painted with a black and white speckle pattern. Tracking and tracing of features on the wing surface is used to recover rigid wing kinematics and deformation of the entire wing surface throughout the wing stroke. Measurement results correspond with the characteristics of natural and artificial flapping wing systems in literature. The wing performs a figure-of-eight motion. The wing exhibits large deformations that vary throughout its stroke, related to both aerodynamic forces and inertial effects. Stroke velocity appears to influence angle of attack as well as spanwise bending, camber and twist.

1 INTRODUCTION

Flapping wing flyers in nature, such as hummingbirds and insects, show impressive flight controllability at high aerodynamic efficiency [1, 2, 3]. These promising aerodynamic characteristics have motivated researchers to develop artificial flapping wing micro and nano aerial vehicles (FWMAV and FWNAV) [4, 5, 6, 7]. Several of these FWAVs are capable of lifting off the ground and performing controlled flight manoeuvres. Flight times vary from a few seconds to several minutes.

Numerical simulations [2, 4, 8, 9] and experimental studies [1, 10, 11, 12] on (artificial) flapping insect wings indicate that both wing kinematics and wing deformation play an important role in aerodynamic force production and aerodynamic efficiency. Investigations into elastic flapping wing structures often study the effect of wing deformation by comparing force production and power consumption to a simulation of an identical rigid wing structure undergoing the same wing motion. The consensus is that spanwise bending, camber and twist have a beneficial effect on aerodynamic performance. Elastic wings outperform rigid wings in situations characterised by high angles of attack, such as insect-like flapping flight [1, 9, 13]. The improved performance is attributed to a more stable attachment of the leading edge vortex (LEV), which prevents flow separation [4, 9, 14]. Percin et al. [11] found that wing deformation is necessary for the wings to benefit from lift enhancing phenomena such as wake capture.

Several experiments have captured the wing motion and deformation of natural and artificial flapping wings. Camera experiments on natural flyers often involve manually tracking wing features frame by frame [15, 16, 17]. This is a time consuming process, difficult for use in sensitivity studies [6, 16]. Other studies draw markers, usually a few dozen [6, 10], onto the wing surface and track them either manually or by (semi-)automatic computer tracking algorithms [18].

This study applies a speckle pattern to the wing surface and tracks wing features through feature detection. Use of a stereoscopic camera setup allows for a three dimensional reconstruction of wing kinematics and deformation. Visualising the wing kinematics and wing deformation may inform improvements in driveline design and wing design. Improved design may result in higher lift capacity, increased flight endurance and flight control.

2 MATERIALS AND METHODS

This work uses an experimental investigation of wing kinematics and wing deformation on an artificial insect-like flapping wing. Figure 1 shows the artificial wing. The wing membrane is a 10 μm Mylar sheet. A carbon fibre skeleton provides structural stiffness. The carbon fibre skeleton consists of four veins: a leading edge (LE), root edge (RE) and two inner veins (R1 and R2). Table 1 lists the wing param-

*Email address(es): thomas.roelandt@kuleuven.be

ters. Figure 2 shows the flapping wing driveline. The wing is actuated through a direct drive [7]. A gear transmission connects the wing to a brushless DC motor that drives the stroke motion. A rubber band positioned at the wing root generates a restoring torque that tries to rotate the wing towards a zero inclination angle. The inclination motion is passive and is the result of an equilibrium between aerodynamic forces and torque from the rubber band. The wing rotates around its first carbon fibre vein. The rubber band at the wing base ensures a positive angle of attack, so this first carbon vein always corresponds to the leading edge. Torsion springs are attached at the wing base. When the wing stroke angle increases, the torsion springs store elastic energy, which is converted and released as kinetic energy at the start of the next stroke. This way the torsion springs assist in stroke reversal. The spring stiffness is chosen such that the natural frequency of the wing driveline is close to the stroke frequency of 20Hz. The resulting resonance is beneficial to driveline efficiency [5, 7, 15]. Stroke frequency and stroke amplitude are determined through a control circuit that consists of a 'Seed Studio XIAO nRF52840 Sense' processing unit and a printed circuit board that was designed in-house. A LiPo battery supplies power to the control circuit and motor.

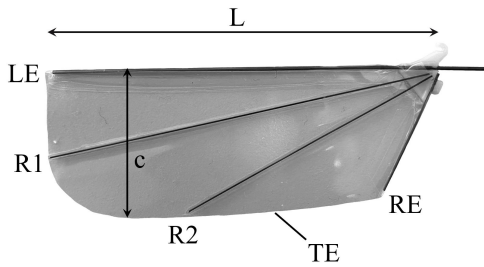


Figure 1: The wing has a skeleton with four carbon fibre rods attached at the wing base. Two outer rods form the wing leading edge (LE) and root edge (RE). Two inner rods (R1 and R2) add stiffness to the wing. A Mylar sheet covers the carbon fibre skeleton and forms the wing membrane. L is the length of the wing membrane, measured along the leading edge. The chord length c is defined as the distance from the leading edge to the trailing edge (TE), measured perpendicular to the leading edge.

2.1 Experimental setup

Figure 2 shows the experimental setup. The setup uses two cameras (Ximea xiB-64 BC120RGCM-X8G3 and JAI SP-12000M-CXP4) that capture wing motion. The transparent wing membrane is covered with a layer of white spray paint and a black speckle pattern of approximately 800 dots with a diameter of approximately 1 mm. Speckle patterns have been used in the past to determine insect-like flapping wing characteristics. Wu et al. [18] used a speckle pattern flapping wing in vacuum to distinguish between inertial and aerodynamic contributions to wing deformation. Doan et al.

[19] used a speckle pattern wing to perform modal analysis on an insect-like wing structure. The effect of spray paint on wing inertia and wing structural properties is negligible. The painting process increases wing mass by 0.012 grams or approximately 5%.

2.2 Motion tracking

The camera image acquisition is synchronised by a trigger unit. Frame rate is set to 800 Hz. At a stroke frequency of 20Hz, 40 images are taken per stroke cycle or 20 images per stroke. Exposure time is set to 100 μ s to limit motion blur. The measurement is repeated with the driveline in various orientations with respect to the cameras to capture the full wing motion pattern. The two camera setup triangulates the coordinates of feature points on the wing membrane throughout wing stroke in a fixed reference frame $x_0y_0z_0$ connected to the driveline (cfr. Figure 3a). Image processing is done in MATLAB R2022a.

Parameter	Symbol	Value	Unit
Wing length	L	73.5	mm
Single wing span	R	83.5	mm
Mean chord length	$\bar{c}$	20.0	mm
Aspect ratio	$\mathcal{A}$	4.17	-
Single wing surface area	S	1670	mm ²
LE diameter	d_{LE}	0.64	mm
R1 diameter	d_{R1}	0.28	mm
R2 diameter	d_{R2}	0.28	mm
RE diameter	d_{RE}	0.42	mm
Membrane thickness	t	10	μ m
Wing mass	m	0.225	g

Table 1: Wing parameters.

2.3 Wing kinematics

Rigid wing motion requires three angular parameters: stroke ϕ , deviation δ and inclination α . Figure 3a illustrates the wing kinematics. Wing stroke is a periodic rotation around z_0 . In studies of birds and insects, these two strokes are classed as an upstroke and downstroke respectively [15]. Previous in-house experiments show that wing motion is symmetric [7]. Therefore there is no need for a distinction between upstroke or downstroke, so only one stroke is analysed. Wing deviation is the wing motion out of the body horizontal plane x_0y_0 . The deviation motion determines whether the wing follows a planar [2, 8, 9, 14, 20], elliptical [10, 14, 21], figure-of-eight [6, 7] or double figure-of-eight [7] profile. All rotations are defined as rotations around positive coordinate axes, so a negative deviation angle corresponds to a wing that is tilted upward. Finally inclination is the wing rotation around its leading edge.

Determining stroke, deviation and inclination from tracked point data is relatively straightforward. The wing rotates around its leading edge, so points $\mathbf{p}$ on the leading

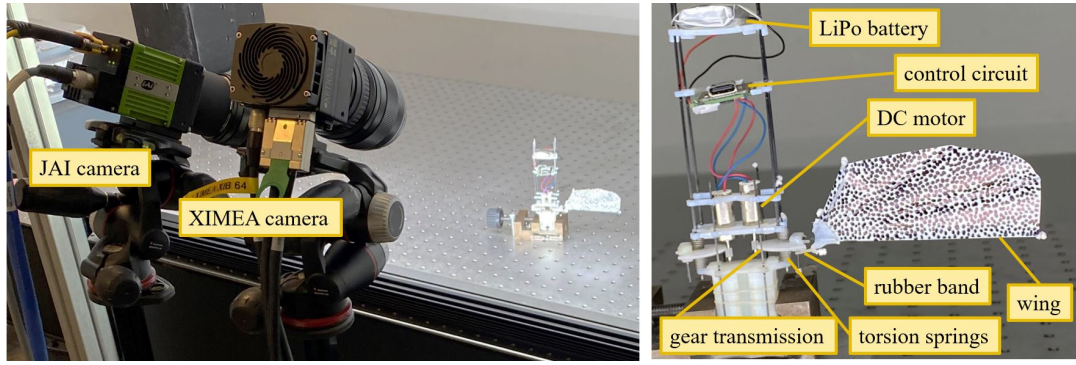


Figure 2: Experimental setup (left), with close-up of driveline and speckle pattern wing (right)

edge have the highest value p_{z0} relative to their distance from the centre of rotation (CoR), according to Figure 3 and with $\mathbf{p} = (p_{x0}, p_{y0}, p_{z0})^T$. A regression line is drawn through a selection of detected points with the highest value for p_{z0} relative to their distance from the CoR. This line is identified as the leading edge. During operation, points on the leading edge undergo stroke motion and deviation motion. Inclination motion does not affect their position. After stroke and deviation, a point at a distance r along the leading edge has coordinates:

$$R_z(\phi)R_y(\delta) \begin{bmatrix} r \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} r \cos \phi \cos \delta \\ r \sin \phi \cos \delta \\ -r \sin \delta \end{bmatrix} \quad (1)$$

$R_z(\phi)$ and $R_y(\delta)$ are the rotation matrices related to stroke and deviation respectively. From coordinates of a point $\mathbf{p}$ on the leading edge, the stroke angle and deviation angle are found:

$$\begin{cases} \phi = \arctan\left(\frac{p_{y0}}{p_{x0}}\right) \\ \delta = \arcsin\left(\frac{-p_{z0}}{\sqrt{p_{x0}^2 + p_{y0}^2 + p_{z0}^2}}\right) \end{cases} \quad (2)$$

To determine inclination, a rigid wing plane is fitted through the marker coordinate data. The rigid wing plane corresponds to the $x_c z_c$ -plane, which contains the wing membrane when no wing deformation is present. A subset of tracked markers closest to the wing CoR is used to fit the rigid wing plane. Points close to the CoR are best suited for this fit, since close to the wing centre the effect of wing deformation is expected to be minor. The inclination angle is then the angle between the rigid wing plane and a vertical plane that contains x_c .

2.4 Wing deformation

Three components together describe wing deformation: spanwise bending, camber (chordwise bending) and twist (torsion). Figure 3b illustrates each deformation component.

The wing membrane is divided into wing chords. Chords are thin strips of wing membrane oriented perpendicular from

the LE to the TE, as shown in Figure 3b. Tracked points are assigned to wing chords based on their distance from the CoR, measured along the LE. For each chord, the LE and TE are computed based on the average of a subset of the three highest and three lowest detected points within that chord.

Spanwise bending q_{sb} is the deformation of the leading edge perpendicular to the rigid wing plane. For each chord, spanwise bending corresponds to the y_c -value of its LE coordinates. Positive spanwise bending means the LE lags behind the rigid wing plane. Negative spanwise bending means the LE is ahead of the rigid wing plane.

Camber q_c is the distance between a point on the wing membrane and the wing chord line. Positive camber means the point on the wing membrane lags behind the rigid wing plane. Negative camber means the point is ahead of the rigid wing plane.

Twist θ_{tw} is the angle between the rigid wing plane and a straight line drawn from LE to TE. Positive twist means the wing chord is pitched down with respect to the rigid wing plane, so the angle of attack is lower compared to the case of a rigid wing. Negative twist means that the wing chord is pitched up with respect to the rigid wing plane, so the angle of attack is higher compared to the case of a rigid wing.

Per wing chord there is a single value for spanwise bending, a single value for twist and camber varies along the chord.

3 RESULTS

3.1 Kinematics measurements

Figure 4 and Figure 5 show the measured wing kinematics. Figure 4 shows the evolution of the deviation angle and the inclination angle throughout wing stroke. Figure 5 shows the wing tip path. The wing motion has a stroke amplitude of 120°. The deviation angle ranges from -10° to 8°. Based on a polynomial fit, the mean deviation angle is equal to -3°. So on average the wing is tilted upward. Mean inclination angle is equal to 29° and maximum inclination angle is equal to 50°.

- At the start of wing stroke, the wing has a high angle of

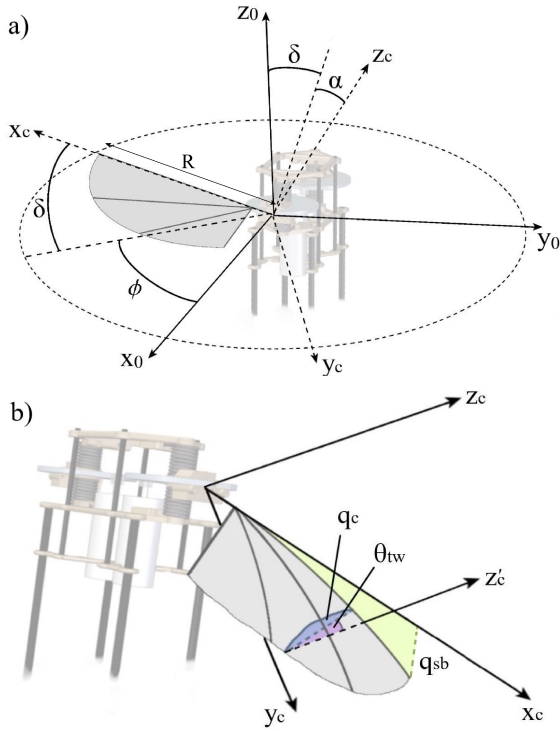


Figure 3: a) Illustration of wing kinematic parameters. Wing kinematics consist of three rotations, expressed with respect to a fixed coordinate system $x_0 y_0 z_0$. The centre of $x_0 y_0 z_0$ corresponds to the wing centre of rotation. x_0 is parallel to the wing when it is at midstroke. $x_0 y_0$ spans the body horizontal plane. $x_c y_c z_c$ follows the wing motion. The stroke angle ϕ is a rotation of the wing inside the body horizontal plane, around z_0 . The deviation angle δ is a rotation around the rotated y-axis. Deviation is the wing out-of-plane motion. The inclination angle α is a rotation around the x_c -axis. The inclination angle determines the angle of incidence. At zero inclination angle, the wing surface is parallel to z_0 . R is the single wing span, measured along the leading edge from the wing centre of rotation to the wing tip. b) Illustration of wing deformation parameters. $x_c z_c$ corresponds to the rigid wing plane. Spanwise bending is defined as the leading edge deformation perpendicular to the wing plane, along y_c . The twist angle θ_{tw} is the angle between $x_c z_c$ and a chord line that connects the local leading edge and trailing edge. Camber q_c is the distance to the local wing chord, measured perpendicular to the chord line going from leading edge to trailing edge.

attack and moves downward. The high angle of attack is approximately constant until midstroke ($\phi = 0^\circ$).

- As the wing approaches midstroke, the wing moves upward.
- After midstroke the wing pitches down, so the angle

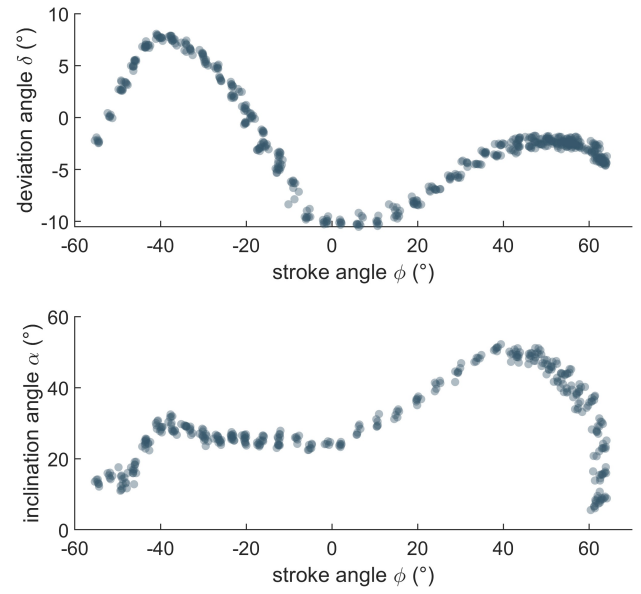


Figure 4: Evolution of a) deviation angle and b) inclination angle throughout stroke. The wing moves from left to right.

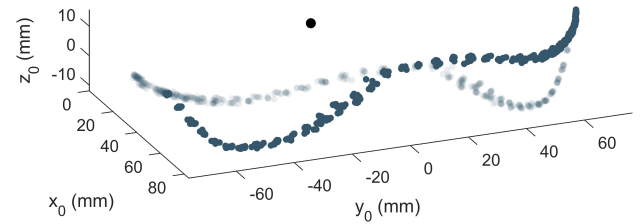


Figure 5: Bold dots show the wing tip path, moving from left to right. The data points are mirrored (transparent dots) with respect to $y_0 z_0$ to clarify the figure-eight motion profile.

of attack decreases. The decrease in angle of attack is likely due to stronger aerodynamic forces acting on the wing membrane, caused by high stroke velocity.

- Near the end of stroke, the wing stroke velocity decreases. The wing pitches back towards zero inclination angle. The wing also moves back to its deviation angle from the beginning of stroke.

Figure 6 shows a side view of the wing motion from start of stroke ($\hat{t} = 0$) to end of stroke ($\hat{t} = 1$). This figure confirms that the inclination angle is close to zero at the start of stroke, then remains small during the first half of stroke. The inclination angle is largest after midstroke and returns to zero at end of stroke. The wing initially dips down, then tilts upward as it nears midstroke and comes back down near the end of stroke.

3.2 Deformation measurements

Figure 7 shows the evolution of each wing deformation component throughout wing stroke.

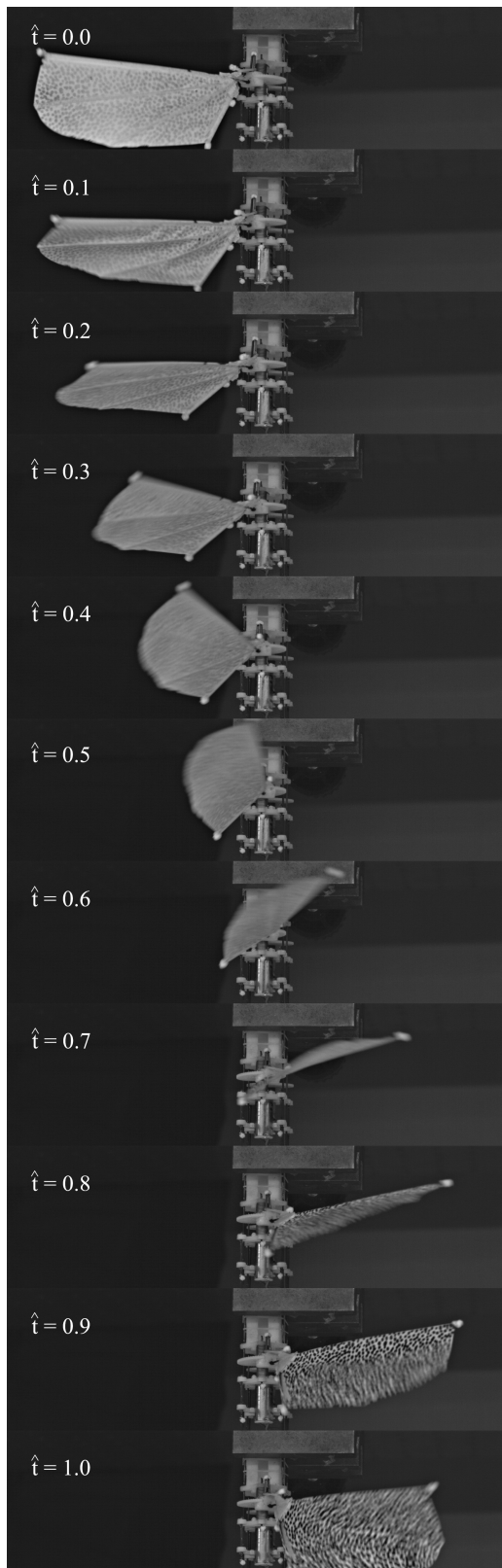


Figure 6: Side view of wing motion.

- At the start of stroke ($\phi = -47.9^\circ \dots -23.2^\circ$), inertial effects dominate wing deformation. While the wing does not yet experience a high stroke velocity, there are significant spanwise bending, negative camber and high twist values close to the wing tip. The leading edge is set in motion, while the rest of the wing membrane appears to lag behind, causing the strong deformation that is also visible in Figure 6 at $\hat{t} = 0.1$. The wing moves at low velocity, yet spanwise bending is significant. Over the wing span, bending increases gradually from 0 mm at the wing root to 3.7 mm at the wing tip. There is simultaneously a large positive camber present between the root edge and inner vein R1, and a large negative camber between R1 and the leading edge. The maximum negative camber is -3.2 mm. Relative to the mean chord length, this equates to $q_c/\bar{c} = -16\%$. The wing experiences a high twist that increases along the wing span up to 26° .
- Towards midstroke ($\phi = -16.2^\circ \dots 10.5^\circ$), the effect of aerodynamic forces acting on the wing membrane becomes more prominent. This is reflected by the strong positive camber, up to 2.7 mm or $q_c/\bar{c} = 13\%$. The largest camber is situated between R1 and the leading edge. Spanwise bending appears rather steady and remains concentrated toward the wing tip. Twist values are lower than at the start of stroke. Maximum twist is at the wing tip, at 19° .
- Deformation after midstroke ($\phi = 24.0^\circ \dots 33.2^\circ$) is smaller than before midstroke. Both spanwise bending and camber values decrease, likely because of the increased inclination angle that lowers the wing angle of attack and thereby aerodynamic forces acting on the wing membrane. Twist values increase on the outer half of the wing, with a maximum of 23° at the wing tip. The increase in wing tip twist is expected. Near the wing tip structural stiffness is low, so the wing has a low resistance to bending. At the same time stroke velocity is highest at the wing tip, causing high aerodynamic forces. The combination of high aerodynamic forces and low stiffness should indeed result in an increased wing tip deformation.
- As the wing nears the end of stroke ($\phi = 47.9^\circ \dots 54.9^\circ$), stroke velocity decreases. At the same time inclination angle is high, so angle of attack is relatively low compared to the first half stroke (cfr. Figure 4). Lower stroke velocity and lower angle of attack result in small deformation.
- At the very end of stroke ($\phi \approx 63^\circ$), the wing rapidly decelerates and inertial effects again take over. This is reflected by negative spanwise bending and a large region of negative camber. The leading edge and wing membrane are ahead of the rigid wing plane.

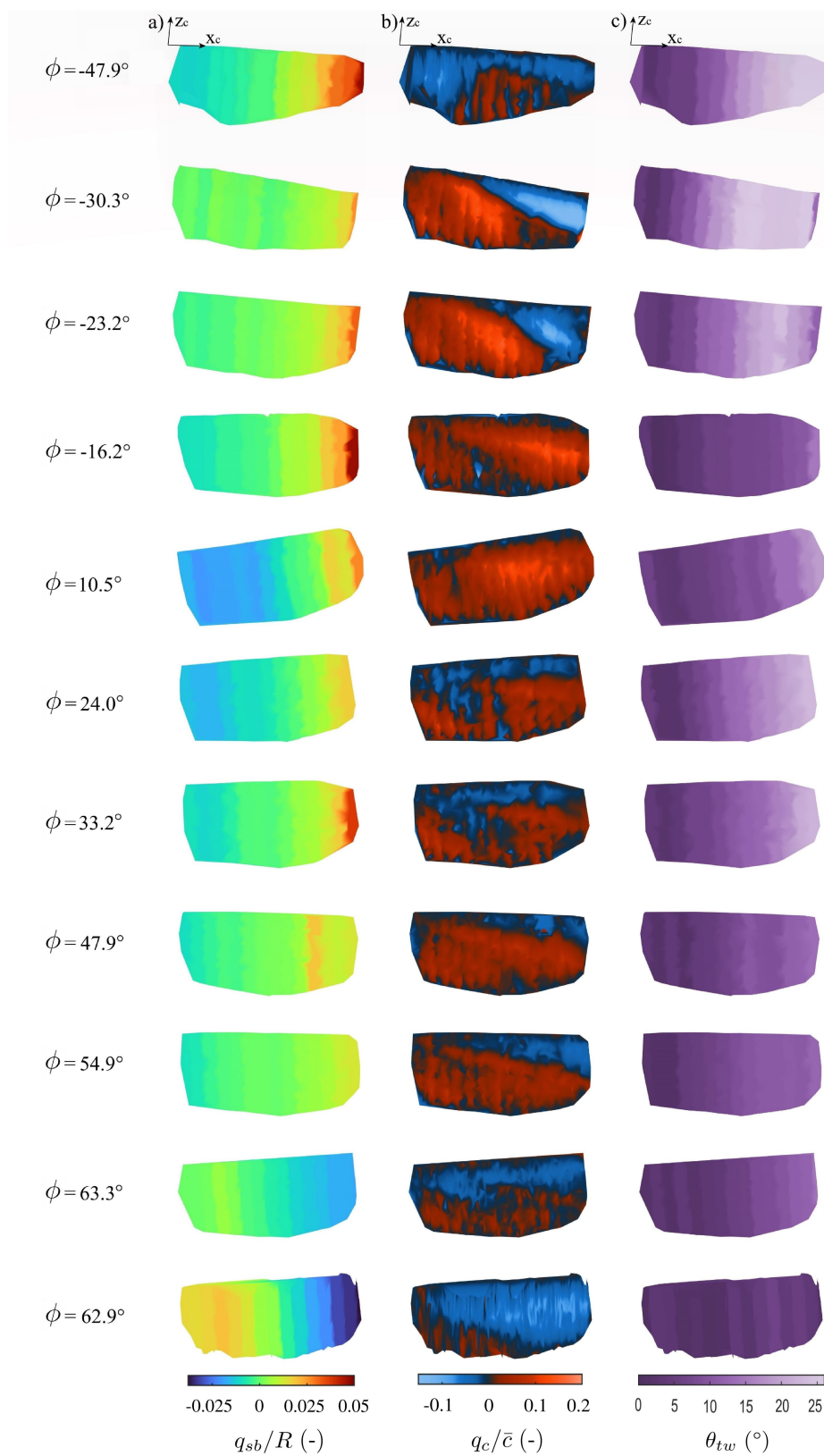


Figure 7: Evolution of deformation components throughout stroke: a) Nondimensional spanwise bending; b) Nondimensional camber; c) Twist, in chronological order from top to bottom.

4 DISCUSSION

The measured wing kinematics follow a logical evolution that agrees with findings from literature [6, 7, 16]. Roshanbin et al. [6] measured a similar wing tip trajectory for their flapping wing mechanism, where the wing initially appears to strongly deviate down, then comes back up toward midstroke and returns to its mean value near end of stroke. Maeda et al. [16] measured the same motion profile on a hummingbird in hovering flight. Previous in-house measurements by Timmermans [7] show a double figure-of-eight pattern when using the same driveline mechanism at a lower stroke amplitude. The wing exhibits a large initial oscillation followed by a second oscillation at lower amplitude, creating a double figure-of-eight pattern. Aerodynamic forces acting on the wing membrane may explain this change in deviation motion profile. At higher stroke velocity, the increase in aerodynamic force acting on the wing membrane prevents the wing from deviating downward a second time.

Wu et al. [18] measured total structural deformation of an artificial flapping wing reinforced with internal veins, which is closest to the wing design that is used in this study. Although they did not separate deformation contributions into spanwise bending, camber and twist, a comparison can still be made. The wing deformation in this work follows a very similar evolution to that measured by Wu et al. Along the leading edge, the out-of-plane deformation increases gradually on the part of the wing that is closest to the wing root and then increases more rapidly towards the wing tip. As stroke velocity increases, deformation related to camber and twist is highest at the trailing edge near the wing tip, where the structural stiffness is lowest. Wu et al. measured a total deformation of 18% compared to chord length. The overshoot related to inertial effects at the end of stroke also agrees with the findings of Wu et al. Camber values between 10% and 25% of mean chord length are common in elastic insect-like flapping wing systems [13, 16, 17, 21]. Camber increases along the wing span. A maximum in positive camber is observed near midstroke. All these observations in literature agree with the findings from the previous section. Hummingbirds and artificial flapping wing systems without internal venation experience higher twist values, with maxima up to 60° [10, 15, 16].

The measurements from the previous section successfully characterise individual wing kinematic components and deformation components. Measurement values and profiles correspond to findings from literature. The experimental setup and processing methodology are suited for additional investigations into wing kinematics and deformation. Future research efforts should explore the effects of wing actuation parameters such as stroke amplitude and stroke frequency and wing design parameters such as vein diameter and venation pattern on wing deformation and aerodynamic performance. Also, the wing structural resonance frequency should be considered during the wing design process, as flapping wings

appear more efficient when actuated close to their structural resonance frequency [4, 8]. Insights from such extensive investigations should prove useful for the wing design process and lead to improved lift production and power requirements of FWAVs. Experiments should be performed under hovering conditions as well as conditions that mimic forward flight, in order to reveal the effects of an additional incoming airflow on wing kinematics and wing deformation. Elastic wing models offer opportunities in the construction of flight controllers that account for the effects of wing deformation. Such models may be especially useful to FWAVs with passive inclination motion, as changes in driving parameters such as stroke amplitude influence or manoeuvres such as forward flight also affect the inclination motion. This can affect aerodynamic force production in a non-obvious way [3]. The influence of a control action or flight manoeuvre on force production is therefore especially difficult to predict in FWAVs with passive wing motion.

5 CONCLUSION

This work uses a stereoscopic high-speed camera setup to track wing motion and deformation of an artificial insect-like flapping wing mechanism. The wing tip undergoes a periodic figure-of-eight motion. Wing inclination is largest after midstroke, when stroke velocity is highest. The wing exhibits large deformation throughout the stroke. When stroke velocity is high, aerodynamic forces likely dominate the wing deformation. At and around stroke reversal, inertial effects are more prominent. Both the kinematics and deformation correspond well with findings from literature.

The large deformations are expected to have a strong effect on aerodynamic force production and aerodynamic efficiency. Follow-up experiments should be performed that include measurements of lift and power consumption. A sensitivity analysis may be performed into the effect of actuation and wing design on wing kinematics and wing deformation and linked with lift production and power consumption. Insights gathered during additional experimental campaigns may lead to improved wing designs with better aerodynamic performance and to a low-computational cost aerodynamic force production model that accounts for wing deformation.

ACKNOWLEDGEMENTS

The Research Foundation - Flanders (FWO) is gratefully acknowledged for its support through research grant numbers G095120N and 1S60222N.

Internal Funds KU Leuven are gratefully acknowledged for their support.

The control circuit that drives wing stroke motion was designed by Tom Henskens, KU Leuven, Celestijnenlaan 300 box 2420, Leuven, Belgium.

REFERENCES

- [1] J. Young, S.M. Walker, R.J. Bomphrey, G.K. Taylor, and A.L.R. Thomas. Details of insect wing design and deformation enhance function and flight efficiency. *Science*, 325:1549–1552, 2009.
- [2] T. Engels, H.N. Wehmann, and F.O. Lehmann. Three-dimensional wing structure attenuates aerodynamic efficiency in flapping fly wings. *Journal of the Royal Society Interface*, 17, 2020.
- [3] A. Menzer, Y. Ren, J.C. Guo, B.W. Tobalske, and H.B. Dong. Wing kinematics and unsteady aerodynamics of a hummingbird pure yawing maneuver. *Biomimetics*, 7(115), 2022.
- [4] H. Liu, S. Ravi, D. Kolomenskiy, and H. Tanaka. Biomechanics and biomimetics in insect-inspired flight systems. *Philosophical Transactions of the Royal Society B - Biological Sciences*, 371, 2016.
- [5] H.V. Phan and H.C. Park. Insect-inspired, tailless, hover-capable flapping-wing robots: Recent progress, challenges, and future directions. *Progress in Aerospace Sciences*, 111(100573), 2019.
- [6] A. Roshanbin, F. Abad, and A. Preumont. Kinematic and aerodynamic enhancement of a robotic hummingbird. *AIAA Journal*, 57(11):4599–4607, 2019.
- [7] S. Timmermans. *Flight performance of nano aerial vehicles with flapping wings: Experiment-based system modelling*. PhD thesis, KU Leuven, Leuven, Belgium, June 2021.
- [8] D. Kumar, G. Singh, P.M. Mohite, E.M. Lau, and Y.-C. Wang. Role of flexibility on the aerodynamic performance of a resonating hummingbird-inspired wing. *The Aeronautical Journal*, 127(1308):193–212, 2023.
- [9] S. Tobing, J. Young, and J.C.S. Lai. Effects of wing flexibility on bumblebee propulsion. *Journal of Fluids and Structures*, 68:141–157, 2017.
- [10] H. Tanaka, H. Suzuki, I. Kitamura, M. Maeda, and H. Liu. Lift generation of hummingbird wing models with flexible loosened membranes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.
- [11] M. Percin, Y. Hu, B.W. van Oudheusden, B. Remes, and F. Scarano. Wing flexibility effects in clap-and-fling. *International Journal of Micro Air Vehicles*, 3(4), 217–227.
- [12] L. Zhao, Q. Huang, X. Deng, and S.P. Sane. Aerodynamic effects of flexibility in flapping wings. *Journal of The Royal Society Interface*, 7(44):485–497, 2009.
- [13] A. Gehrke, J. Richeux, E. Uksul, and K. Mulleners. Aeroelastic characterisation of a bio-inspired flapping membrane wing. *Bioinspiration & Biomimetics*, 17(6), 2022.
- [14] W. Shyy, H. Aono, S.K. Chimakurthi, P. Trizila, C.-K. Kang, C.E.S. Cesnik, and H. Liu. Recent progress in flapping wing aerodynamics and aeroelasticity. *Progress in Aerospace Sciences*, 46:284–327, 2010.
- [15] R. Ingersoll and D. Lentink. How the hummingbird wing is tuned for efficient hovering. *Journal of Experimental Biology*, 221(20), 2018.
- [16] M. Maeda, T. Nakata, I. Kitamura, H. Tanaka, and H. Liu. Quantifying the dynamic wing morphing of hovering hummingbird. *Royal Society Open Science*, 4(9), 2017.
- [17] S.M. Walker, A.L.R. Thomas, and G.K. Taylor. Deformable wing kinematics in the desert locust: how and why do camber, twist and topography vary through the stroke? *Journal of The Royal Society Interface*, 6:735–747, 2009.
- [18] P. Wu, B. Stanford, W. Bowman, A. Schwartz, and P. Ifju. Digital image correlation techniques for full-field displacement measurements of micro air vehicle flapping wings. *Experimental Techniques*, 33:53–58, 2009.
- [19] N.V. Doan, V.T. Le, H.C. Park, and N.S. Goo. Modal analysis using a virtual speckle pattern based digital image correlation method: An application for an artificial flapping wing. *Experimental Mechanics*, 62(2):253–270, 2022.
- [20] J. Choi, D. Gong, J. Lee, C. Kim, and S. Shin. Simulation of the flapping wing aerial vehicle using flexible multibody dynamics. *International Journal of Micro Air Vehicles*, 13, 2021.
- [21] Y. Dong, B. Song, D. Xue, and W. Yang. 3d numerical simulation of a hovering hummingbird-inspired flapping wing with dynamic morphing. *Journal of Applied Fluid Mechanics*, 15(3):873–888, 2022.

Design and Path Following Flight Test of a Flapping-Wing Air Vehicle

Sang-Gil Lee, Hyeon-Ho Yang, Hwankyun You, and Jae-Hung Han*

Korea Advanced Institute of Science and Technology, 291 Daehak-ro, Yuseong-gu, Daejeon, Republic of Korea

ABSTRACT

Flapping-wing air vehicles (FWAVs) have significant advantages in surveillance and reconnaissance missions due to their camouflage capabilities. In such missions, autonomous flight capabilities in outdoor environments are important. An FWAV capable of autonomous flight is designed in this research. The FWAV has elevons for generating pitching and rolling moments and can carry essential payloads such as the flight control computer (FCC) and GPS for outdoor autonomous flights. The flight test result demonstrates that the designed FWAV can autonomously follow a given circular path with implemented control method.

1 INTRODUCTION

Flying animals such as birds and insects have an excellent flight capability, allowing them to fly in forests and indoor environments. They have outstanding flight time and maneuverability compared to conventional aircrafts of similar size. These flying animals fly by flapping their wings, bio-inspired flapping-wing air vehicles (FWAVs) mimic this flight mechanism to pursue the flight capability of flying animals. Since FWAVs do not have a propeller rotating at high speed, they have many advantages, such as low noise and the ability to fly close to people. FWAVs also have the potential advantage that they can utilize various flight modes of flying animals, such as soaring, perching, and flap-gliding. In addition to these advantages, FWAVs are well-suited for surveillance and reconnaissance missions due to their excellent camouflage capabilities.

Insect-type FWAVs are advantageous for indoor environments with their hovering capabilities, and various studies have been conducted about their interesting aerodynamic [1, 2] and aeroelastic characteristics [3, 4]. On the other hand, bird-type FWAVs are more advantageous for outdoor missions in terms of payload capability and disturbance sensitivity than insect-type FWAVs. E-flap [5] is a bird-type FWAV developed with a focus on payload capability. The vehicle has achieved a payload capability of 520 g, which is close to its own mass. Meanwhile, SmartBird [6] focuses on mimicking the wing motion of real birds. It has multiple joints, so it

can decrease downward force by reducing the wing area during the upstroke and can realize twist motion in the spanwise direction through an additional actuator.

The two above-mentioned FWAVs successfully achieved their goals, which are payload capacity and mimicking wing motion. However, the flight tests for those vehicles were conducted with manual control. Although autonomous flight has been performed in an indoor environment with small disturbance, the vehicle's positioning relied on an external device [7, 8]. Various control methods for FWAV have been proposed, but the number of studies that have successfully implemented control methods for outdoor flight is limited.

Dove [9] is capable of autonomous outdoor flight. Dove features a customized lightweight flight control system. The reason is that, considering Dove's specifications with a mass of 220 g and a wingspan of 50 cm, it is judged that the general electronic devices designed for unmanned aerial vehicles (UAVs) are not suitable in terms of their payload capability.

The vehicle developed in this research is focused on autonomous flight in the outdoor mission. It can carry a general flight control system for UAV (Pixhawk 4 mini and its GPS module), the control method for the path following is implemented, and a flight test is conducted to evaluate its performance in following a given path.

2 MATERIALS AND METHOD

2.1 Hardware design

The design mass is set based on the mass of electronics. The main electronics for outdoor autonomous flight are a flight control computer (FCC) and GPS. Pixhawk 4 mini, which is a general FCC for UAVs, and its GPS module are used. The masses of the FCC and GPS are 37 g and 33 g, respectively. In this study, the design mass is set at 700 g so that the electronics account for 10%, referring to the fact that electronics of Dove [9] and E-Flap [5] account for 16% and 11%, respectively. Wingspan can be calculated to be 1 m using $\text{Wingspan} = 0.506 \cdot \text{Weight}^{1/3}$ [10], and 1.2 m is selected to have a margin of 20%.

Before hardware design of the powertrain, it is necessary to determine the required power to select the proper motor for the powertrain. Firstly, wing kinematics is defined with the wing-fixed frame in Figure 1. The flapping angle ϕ refers to the x -axis rotation, and the twisting angle θ refers to the y -axis rotation. The flapping and twisting angles are assumed as follows:

*Email address(es): jaehunghan@kaist.ac.kr

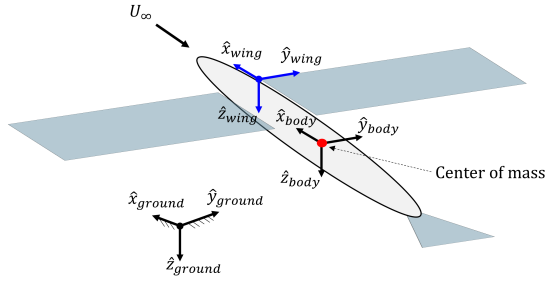


Figure 1: Reference frames of FWAV model

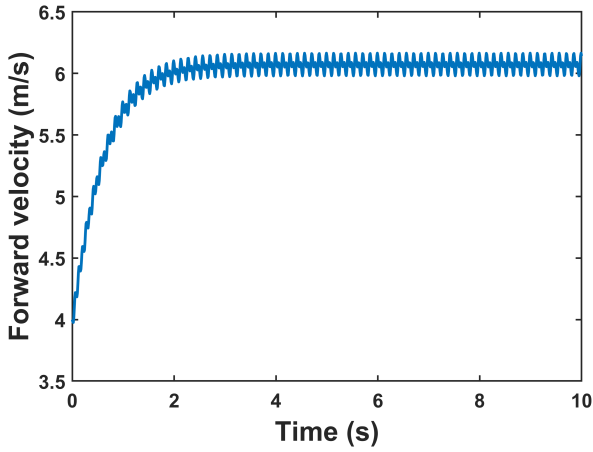


Figure 2: Converged forward velocity (Flapping frequency: 7 Hz, Twisting amplitude: 60°)

$$\phi_w(t) = \frac{\phi_{amp}}{2} \cos(2\pi ft) \quad (1)$$

$$\theta_w(t) = \frac{\theta_{amp}}{2} \sin(2\pi ft) \quad (2)$$

where ϕ_{amp} , θ_{amp} , f , and t represent flapping amplitude, twisting amplitude, flapping frequency, and time, respectively. Although the mechanism to be designed generates only flapping motion, the passive wing twisting induced by the structural deformation is assumed to have a 90-degree phase difference from the flapping motion.

The aerodynamic model used in this study is based on the quasi-steady aerodynamic model [11], and as a result, lift and drag coefficients are given as Equation 3 and 4. α represents the angle of attack in degrees. Each wing is discretized into multiple aerodynamic strips, and the aerodynamic loads on the strips are calculated with the lift and drag coefficients. Aerodynamic loads of each wing can be obtained by summing the lift and drag forces acting on the strips of the each wing.

$$C_L = 0.225 + 1.58 \sin(2.13\alpha - 7.20) \quad (3)$$

$$C_D = 1.92 - 1.55 \cos(2.04\alpha - 9.82) \quad (4)$$

To obtain the required power for the current FWAV with a 1.2 m wingspan, the flight dynamic simulations are conducted. The wing geometry used for analysis is simplified to a rectangular wing with an aspect ratio of 5, the simulation condition is that an pitch angle of body is 15°, and ϕ_{amp} is 45°. If only the x-axis of the ground-fixed frame has a degree of freedom, the forward speed converges to the trim state as represented in Figure 2. At this time, the cycle-averaged lift is calculated.

The results of flight dynamic simulation are shown in Figure 3. Assuming that the twisting amplitude is 40° or more, it can be confirmed that the flapping frequency need to be at least 7 Hz to generate lift greater than the design mass. The required power for flapping motion can be computed by multiplying the moment at the pivot point of the wing by the angular velocity, and the results are shown in the Figure 4. Because the maximum value of power consumption at 7 Hz is 109 W, the output power of motor should be greater than 109 W.

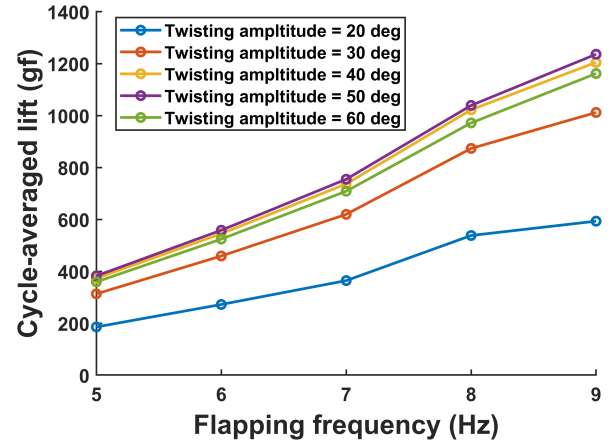


Figure 3: Cycle-averaged lift

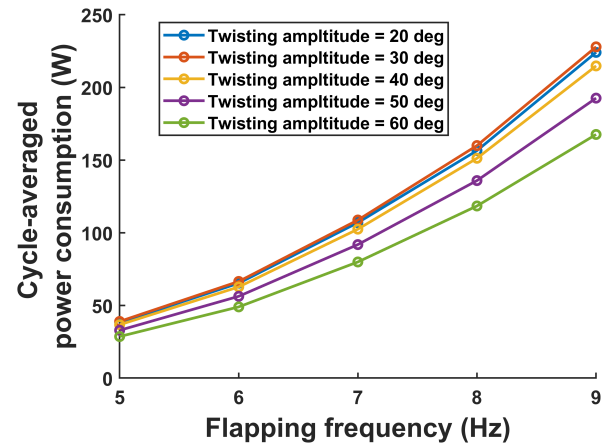


Figure 4: Cycle-averaged power consumption

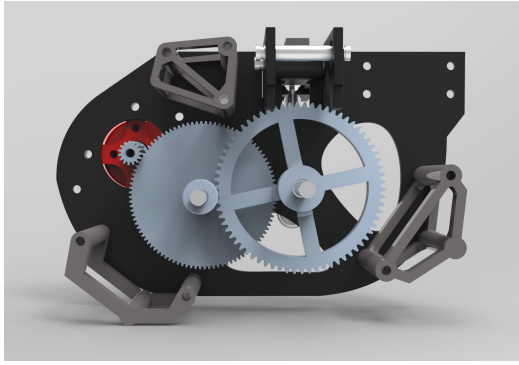


Figure 5: 2-stage gear train

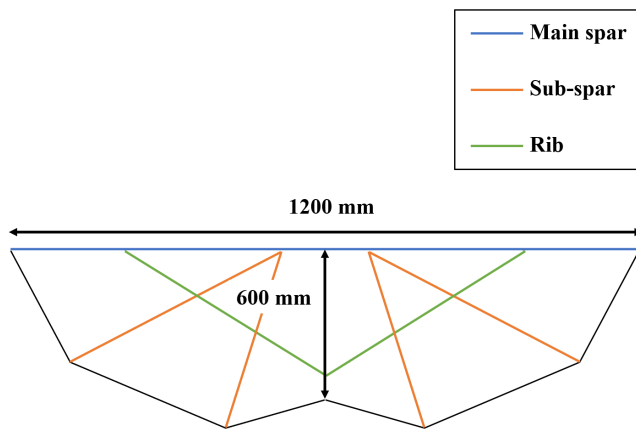


Figure 6: Geometry of main wing

A motor is selected with a margin considering factors such as transmission losses. The selected motor is a 1380 Kv brushless DC motor with a maximum output power of 178 W. The rotation speed of the motor is reduced with a 2-stage gear train. The first stage has a 6.67:1 reduction ratio, and the second stage has a 5.83:1 reduction ratio. The CAD model of the gear train is shown in Figure 5. The shaft of the gear train is supported by two side plates, which are 3D printed with Onyx from Markforged Inc. and represented in black color in Figure 5. Although carbon fiber-reinforced plastic (CFRP) plates can be used for the parts to reduce weight, this study opts for 3D printing for rapid prototyping. As the gear module increases, the size of the gear teeth also increases, enabling the gear to withstand higher contact forces. However, the gear radius also increases, resulting in an overall increase in the size of the mechanism. Because the first stage has a higher rotation speed and lower contact force between teeth than the second stage, the first stage has module 0.5 gears, and the second stage has module 0.8 gears. The output shaft of the gear train drives the revolute-spherical-spherical-revolute (RSSR) mechanism.

The main wing is connected to the RSSR mechanism and

generates lift and thrust. The wingspan has been set to 1.2 as previously mentioned, and the geometry of the wing shape is shown in Figure 6. The coated fabric is used as a material for a wing sheet. The wing frame consists of main spars, sub-spars, and ribs, and it is fabricated using CFRP rods and pipes. The wing frame is attached to the wing sheet using adhesive tape.

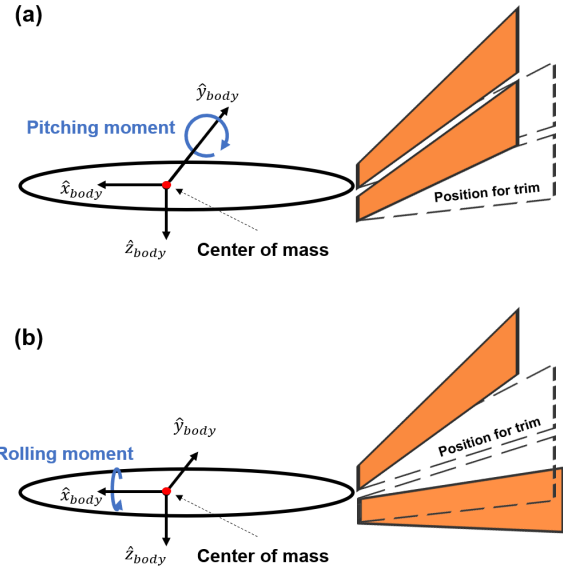


Figure 7: Configurations of elevons to generate control moment: (a) pitching moment; (b) rolling moment.

The tail wing is used for stabilization and control of FWAV. Configurations of control surfaces for FWAV are generally divided into two types: Elevons, and Rudder-elevator. Elevon is a compound word for elevator and aileron. The left and right horizontal control surfaces of the tail wing move independently. Therefore, they can generate pitching and rolling moments as shown in Figure 7. Rudder-elevator type is a common type for conventional aircraft. Pitching and yawing moments can be generated. Attaching control surfaces such as ailerons to the main wing of an FWAV is challenging because that the main wings have a flapping motion. As a result, roll stability relies on passive factors such as dihedral angles and the flapping counter-torque (FCT) [12]. In this study, the elevons are used as a control surface because active control of the roll direction rather than the yaw direction is considered to be more important.

The powertrain and the tail wing are connected using a CFRP pipe. The main wing is connected to the powertrain and the CFRP pipe. The CAD model of designed FWAV is shown in Figure 8. The fabricated FWAV in Figure 9 has a slightly higher mass than the design mass, and its specification is listed in Table 1.



Figure 8: CAD model of flapping-wing air vehicle



Figure 9: Fabricated flapping-wing air vehicle

Parameter	Value
Mass	740 g
Wingspan	1.2 m
Endurance	4 min
Aspect ratio	5.2

Table 1: Specification of flapping-wing air vehicle

2.2 Controller

Path following control is implemented based on the PX4 autopilot [13], open-source software. The overall structure of the control system is shown in Figure 10. The position controller in the total control system, is shown in Figure 11. Because the designed FWAV does not have a control surface for a yawing moment, the position controller only gives pitch and roll commands to the attitude controller, not the yaw command. It is assumed that yaw dynamics have enough stability from the flapping counter torque (FCT). The attitude controller has a cascaded loop structure. In an outer loop, angu-

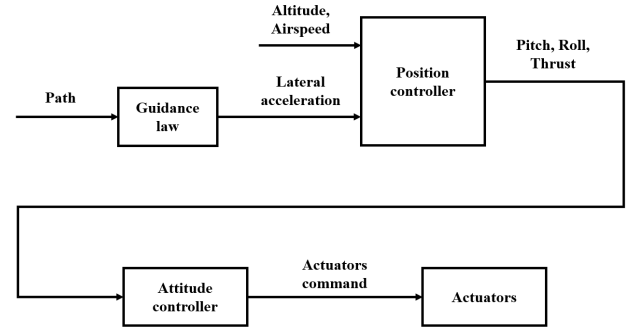


Figure 10: Overall structure of control system

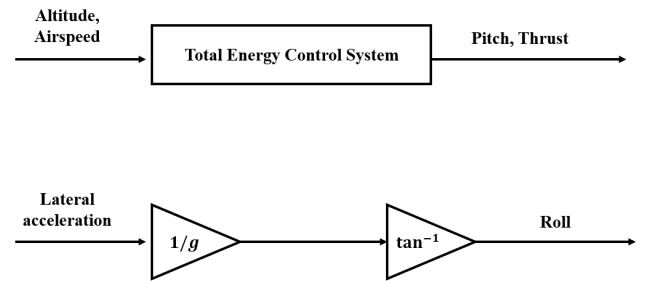


Figure 11: Position controller

lar rate setpoints are generated with attitude error. In an inner loop, the angular rate setpoints are tracked with proportional and integral (PI) feedback controllers and a proportional feed-forward controller. For the details of guidance law and total energy control system, please refer to T. Stanstny and R. Siegwart [14] and A. A. Lambregts [15], respectively.

3 FLIGHT TEST

The flight test was conducted to evaluate the performance of the designed FWAV and validate the implemented control method for autonomous flight. The flight test followed the sequence of takeoff, ascent, path following, descent, and landing. The FWAV was fully autonomously controlled during path following flight. Figure 12 shows a snapshot of conducted flight test, and the FWAV is highlighted with yellow circles.

The path following results are shown in Figure 13 and 14. The reference path was given as the circle with a 25 m radius and a constant altitude. The dashed line indicates the result of manual flight mode. In manual flight mode, only an attitude controller was active to track attitude commands given manually. After switching to the path following mode, the at-



Figure 12: Flight test snapshot

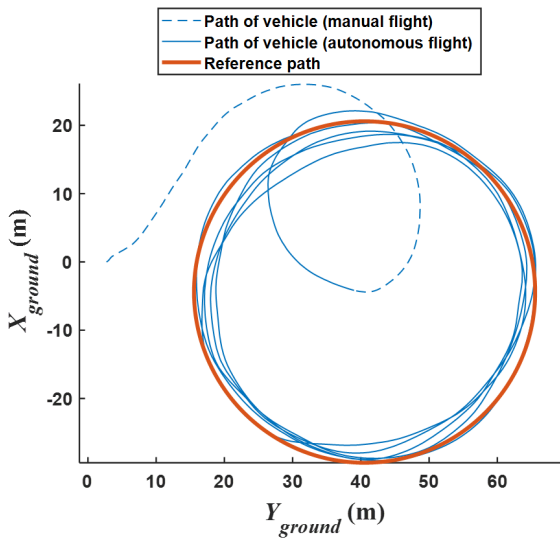


Figure 13: Path in horizontal plane

altitude commands were generated by position controller, and the attitude commands are shown in Figure 15. To follow the circular path in clock-wise direction, the roll command were given as a positive value. By tracking the attitude commands, the implemented controller successfully enabled the FWAV to follow the reference path.

4 CONCLUSION

In this study, the FWAV capable of autonomous flight was designed, and the flight test was performed to validate the im-

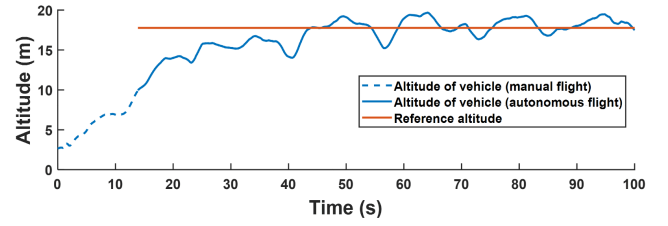


Figure 14: Altitude

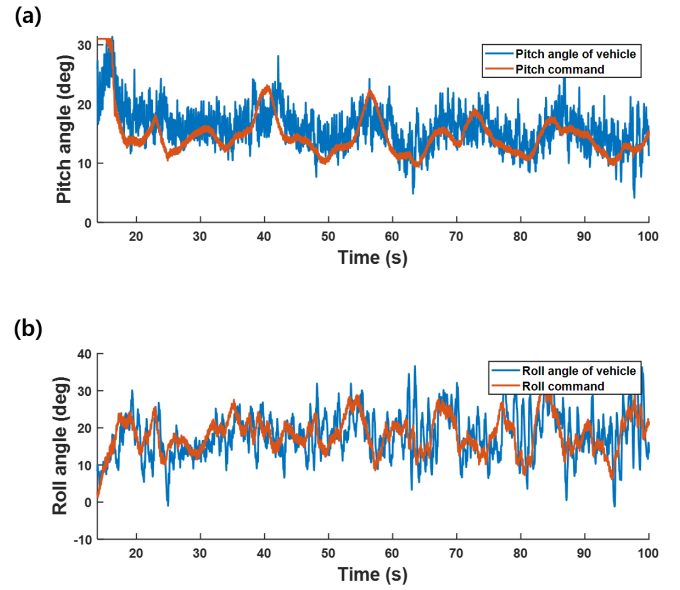


Figure 15: Attitude control result: (a) pitch angle; (b) roll angle.

plemented control method. The powertrain of the FWAV included a BLDC motor, a gear train, and an RSSR mechanism. The design of the powertrain was based on the established dynamics model. Elevons were chosen over the conventional rudder-elevator configuration due to their capability to generate rolling moments. To control the FWAV, which cannot generate a yawing moment, an appropriate control method was implemented based on the PX4 autopilot. The conducted flight tests demonstrated that the implemented control method can be successfully applied to the FWAV, enabling the FWAV to track the desired path.

The FWAV developed in this study has a relatively large size compared to the existing FWAVs designed for autonomous flight, allowing it to accommodate the avionics typically used in general UAVs. As a result, it is advantageous for outdoor missions. The designed FWAV is expected to be used as a test platform to validate the theoretically proposed control theories.

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 (2020M3C1C1A01083415).

REFERENCES

- [1] J.-S. Han, A. T. Nguyen, and J.-H. Han. Aerodynamic characteristics of flapping wings under steady lateral inflow. *Journal of Fluid Mechanics*, 870:735–759, 2019.
- [2] A. T. Nguyen, J.-S. Han, and J.-H. Han. Effect of body aerodynamics on the dynamic flight stability of the hawkmoth *manduca sexta*. *Bioinspiration & biomimetics*, 12(1):016007, 2016.
- [3] R. Addo-Akoto, J.-S. Han, and J.-H. Han. Aerodynamic characteristics of flexible flapping wings depending on aspect ratio and slack angle. *Physics of Fluids*, 34(5), 2022.
- [4] A. T. Nguyen and J.-H. Han. Wing flexibility effects on the flight performance of an insect-like flapping-wing micro-air vehicle. *Aerospace Science and Technology*, 79:468–481, 2018.
- [5] R. Zufferey, J. Tormo-Barbero, M. M. Guzmán, F. J. Maldonado, E. Sanchez-Laulhe, P. Grau, M. Pérez, J. Á. Acosta, and A. Ollero. Design of the high-payload flapping wing robot e-flap. *IEEE Robotics and Automation Letters*, 6(2):3097–3104, 2021.
- [6] W. Send, M. Fischer, K. Jebens, R. Mugrauer, A. Nagarathinam, and F. 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*, volume 10. Brisbane, Australia, 2012.
- [7] H.-Y. Kim, J.-S. Lee, H.-L. Choi, and J.-H. Han. Autonomous formation flight of multiple flapping-wing flying vehicles using motion capture system. *Aerospace Science and Technology*, 39:596–604, 2014.
- [8] H.-Y. Kim, J.-S. Lee, and J.-H. Han. Indoor autonomous flight of ornithopter using motion capture system. *International Journal of Intelligent Unmanned Systems*, 1(3):204–214, 2013.
- [9] W. Yang, L. Wang, and B. Song. Dove: A biomimetic flapping-wing micro air vehicle. *International Journal of Micro Air Vehicles*, 10(1):70–84, 2018.
- [10] T. Liu. Comparative scaling of flapping-and fixed-wing flyers. *AIAA journal*, 44(1):24–33, 2006.
- [11] M. H. Dickinson, F.-O. Lehmann, and S. P. Sane. Wing rotation and the aerodynamic basis of insect flight. *Science*, 284(5422):1954–1960, 1999.
- [12] T. L. Hedrick, B. Cheng, and X. Deng. Wingbeat time and the scaling of passive rotational damping in flapping flight. *Science*, 324(5924):252–255, 2009.
- [13] L. Meier, D. Honegger, and M. Pollefeys. Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6235–6240, 2015.
- [14] T. Stastny and R. Siegwart. On flying backwards: Preventing run-away of small, low-speed, fixed-wing uavs in strong winds. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5198–5205, 2019.
- [15] A. A. Lambregts. Vertical flight path and speed control autopilot design using total energy principles. In *Guidance and Control Conference*, page 2239, 1983.

Integrated Simulation Program for Initial Design of Bird-Inspired Ornithopter

H.-H. Yang, S.-G. Lee, E.-H. Lee, and J.-H. Han*

Korea Advanced Institute of Science and Technology, 291 Daehak-ro, Yuseong-gu, Daejeon, Republic of Korea

ABSTRACT

This study presents an integrated simulation program for the initial design of a bird-inspired ornithopter. The design and analysis of an ornithopter is a challenging task due to the complex interaction between the unsteady aerodynamic loads and wing structural deformation during the flapping flight. This study focuses on the design of a single-joint ornithopter model. The unsteady vortex lattice method (UVLM) is modified and used to predict the aerodynamic loads generated by flapping motion. The commercial software RecurDyn is used for multi-flexible-body dynamics (MFBD) analysis, and a flexible wing model is modeled using beam and shell elements. The wing kinematics of the ornithopter model is based on a planar crank-rocker. The proposed aerodynamic model is validated against the experimental results and the operation procedure of the developed ORNithopter Integrated Simulation Program (ORNISP) is introduced. As an example of the use of ORNISP, the effect of wing flexibility on ornithopter aerodynamics is analyzed.

1 INTRODUCTION

An ornithopter is an aircraft that achieves flight by flapping its wings, mimicking the flight characteristics of biological fliers. At low Reynolds numbers, the flapping motion of an ornithopter results in complex unsteady flows, which are utilized for efficient flight. Various research groups have been developing different types of ornithopters to mimic bird flight characteristics such as rapid maneuverability, camouflage, and efficient flight[1, 2, 3]. Especially, J. Gerdes *et al.*[4] developed the Robo Raven which is a servo-driven ornithopter capable of outdoor aerobatics using independently actuated and controlled wings.

However, the design and analysis of the bio-inspired flapping-wing air vehicle (FWAV) have unique challenges, primarily induced by the interaction between the unsteady aerodynamic loads and wing structural deformation during the flapping flight. Typically, the aerodynamic characteristics of FWAVs are investigated through a computational fluid-

structure interaction (FSI) based on computational fluid dynamics (CFD) and computational structural dynamics (CSD) solvers[5, 6, 7]. One of the most prominent obstacles in the design process is the considerable computational time required by both CFD and CSD solvers. CFD, important for performing unsteady aerodynamic analysis, is notably time-intensive. Similarly, CSD, employed to analyze wing deformation induced by unsteady aerodynamic loads, also requires significant computation time. W. Yang *et al.*[8] developed a bird-inspired ornithopter, named DOVE, using CFD and CSD solvers, but a wing model with good lift and thrust generation was selected by comparing only three different types of flexible wing models. These extensive computational requirements of CFD and CSD often hinder the comprehensive understanding of the aerodynamic characteristics of ornithopters during the initial design stages, particularly when exploring various design parameters such as wing shape and wing kinematics.

Q. Wang *et al.*[9] proposed a computationally efficient FSI model to enable the parametric study and optimization of flapping wing twists and corresponding kinematics. The proposed model enables full FSI analysis in just a few minutes since an analytical twist model and a quasi-steady aerodynamic model are used as the structural and aerodynamic models, respectively. Similarly, R. Schwab *et al.*[10] proposed a low-order FSI model that uses a modal-analysis-based structural model and a quasi-steady aerodynamic model: blade element theory (BET). However, both FSI models have a low fidelity in unsteady aerodynamic analysis because the quasi-steady aerodynamic model, which cannot consider the non-linearity of the wake effect, was applied to a flapping-wing model. In order to improve the fidelity, since the wake induced by flapping motion has a significant effect on unsteady aerodynamics in flapping flight, it is necessary to use an aerodynamic model that can take this into account[11, 12].

In this study, an integrated simulation program for the initial design of a bird-inspired ornithopter, named ORNithopter Integrated Simulation Program (ORNISP), is proposed. The unsteady vortex lattice method (UVLM), which can consider wake-shedding, is modified and implemented to compute the aerodynamic loads produced by the flapping motion. The multi-flexible-body dynamics (MFBD) analysis is carried out by using the commercial software RecurDyn. A flexible wing model is modeled using beam and shell elements and is converted into a modal-based flexible model. The wing kinematics of the ornithopter model is based on a planar crank-rocker

*Email address(es): jaehunghan@kaist.ac.kr

mechanism. The proposed aerodynamic model is validated against experimental results, and the operation procedure of the developed ORNISP is introduced. As an example of the use of ORNISP, the effect of wing flexibility on ornithopter aerodynamics is analyzed.

2 MATERIALS AND METHOD

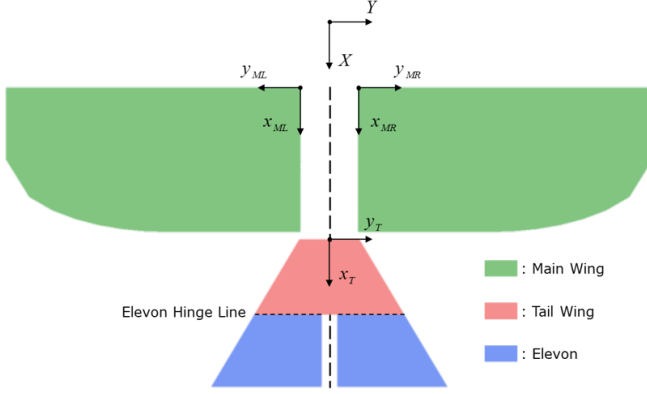


Figure 1: Schematic diagram of ornithopter model

2.1 Ornithopter Model

Figure 1 shows the schematic diagram of the ornithopter model. This study employs a single-joint ornithopter model in which the main wing undergoes flapping motion only about the x -axis of each body-fixed frame (x_{MR} , x_{ML}). As shown in Figure 1, the ornithopter model is composed of two main wings and one tail wing with applied elevons. Each elevon can rotate along the hinge line connected to the tail wing, and its angle can be adjusted to generate control forces for pitching or rolling moments.

2.2 Aerodynamic Model

This study utilizes the unsteady vortex lattice method (UVLM), which is based on potential flow theory, to compute the aerodynamic loads generated by the flapping motion of the ornithopter. In the body-fixed frame that is fixed on the wing, a velocity potential (Φ) can be defined. The flow around the wing is considered incompressible, inviscid, and irrotational, resulting in the continuity equation being expressed as the Laplace equation, represented as:

$$\nabla^2 \Phi = 0. \quad (1)$$

Laplace's equation can be solved using vortex ring elements[13]. By superimposing these vortex ring elements, the complex flows around the moving wing can be simulated.

As depicted in Figure 2, the wing model has chord and span lengths of c and b , respectively, and is divided into M chord-wise and N span-wise panel elements for numerical analysis. Each panel element is assigned i and j indices,

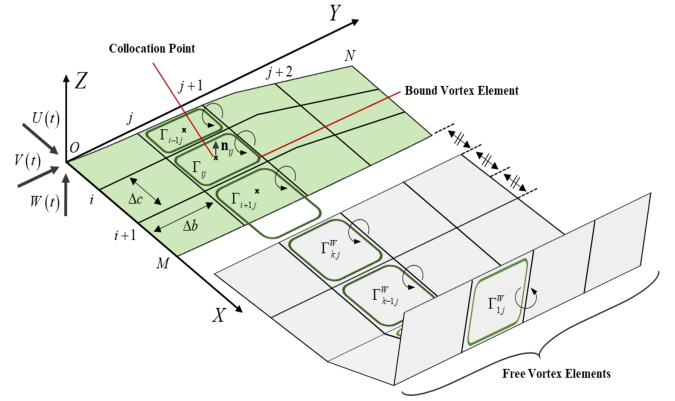


Figure 2: Schematic diagram of modified UVLM

where i denotes the chord-wise index and j denotes the span-wise index of the panel element. A uniform vorticity is assumed on each panel element and replaced with a bound vortex ring of strength Γ_{ij} located at the edge of the panel element. Here, Γ_{ij} represents the vortex strength of the bound vortex ring on the (i, j) th panel element. The middle of each panel element, where the zero normal flow boundary condition is enforced, serves as the collocation point. The normal vector of each panel element is denoted as $\mathbf{n}_{ij}$.

The wakes are simulated using free vortex ring panels of strength Γ_{kj}^w that are shed from the trailing edge of the wing. Here, Γ_{kj}^w represents the vortex strength of the free vortex ring of the k th time step and j th span-wise wake panel element. The shedding of the wake panel element begins at one-third of the distance between the trailing edge and the last free vortex ring. This shedding location is advantageous as it allows for the consideration of not only the wing motion but also the convection of the previous shed vortices[14, 15].

To calculate the pressure difference (Δp_K), a vorticity vector (γ_K) is employed, and the unsteady Bernoulli equation is utilized as follows:

$$\Delta p_K(t) = \rho \left[\{ (U(t) + u_W, V(t) + v_W, W(t) + w_W)_K \times \gamma_K \} \cdot \mathbf{n}_K + \frac{1}{\Delta t} \left(\sum_{K=1}^m \Gamma_K(t) - \sum_{K=1}^m \Gamma_K(t - \Delta t) \right) \right] \quad (2)$$

where ρ is the flow density, and $(U(t), V(t), W(t))_K$ represents the kinematic velocity due to the wing's motion, while $(u_W, v_W, w_W)_K$ represents the induced velocity resulting from the wake panels. Additionally, Δt is the time step size of the simulation, and m is equal to $M \times N$. The aerodynamic loads are determined by integrating all the pressure differences along the surface of the wing.

This study introduces a novel numerical scheme, named the pseudo leading-edge vortex (PLEV) model shown in Figure 3, to account for the effects of flow separation and

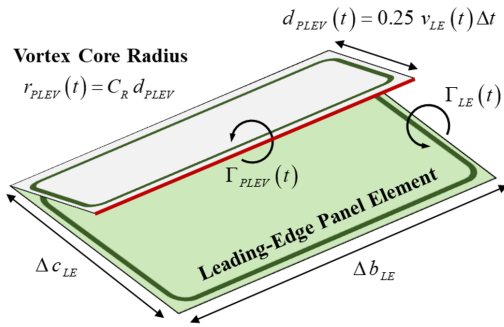


Figure 3: Schematic diagram of PLEV model

leading-edge vortex during flapping flight. The PLEV model creates a PLEV ring panel from each leading-edge panel of the wing. In this case, the trailing PLEV line is created behind the leading-edge by a distance of d_{PLEV} . The strength of the PLEV (Γ_{PLEV}) is affected by the strength of the leading-edge panel (Γ_{LE}) and its local induced velocity (v_{LE}). Moreover, the vortex core radius of the PLEV (r_{PLEV}) is equal to the product of d_{PLEV} and core radius coefficient (C_R).

2.3 Multi-Flexible-Body Dynamics Model

In the ornithopter model, the main wing can be modeled as a rigid wing to simplify the analysis by ignoring the deformation of the wing. However, in order to more accurately analyze the aerodynamic characteristics of the ornithopter, the wing should be modeled as a flexible wing, which requires a multi-flexible-body dynamics (MFBD) analysis for the entire model.

In this study, the commercial software RecurDyn is employed as the solver for the MFBD analysis. At first, a finite element (FE) model of a flexible wing is modeled using beam and shell elements. The beam elements represent the wing frame structures such as spars and ribs in the main wing. The shell elements represent the wing sheet structure to which aerodynamic loads are enforced. This FE flexible wing model is further converted into a modal-based flexible wing by modal analysis since the FE model requires considerable computational time for analysis.

In order to consider the interaction between the aerodynamic model and the structural model, a straightforward coupling method termed the nearest-neighbor interpolation is introduced. In this method, a node in the aerodynamic mesh of the wing (Figure 2) is coupled with the nearest node in the structural mesh (Figure 4 (B)), resulting in both nodes having the same deformation. The aerodynamic force acting on an aerodynamic node is applied at the nearest structural node from the collocation point. The validity and effectiveness of this simple coupling method have been proved in the research conducted by A. T. Nguyen and J.-H. Han[16].

The FSI, which uses the modified UVLM as an analysis method for unsteady aerodynamics and a modal-based flexi-

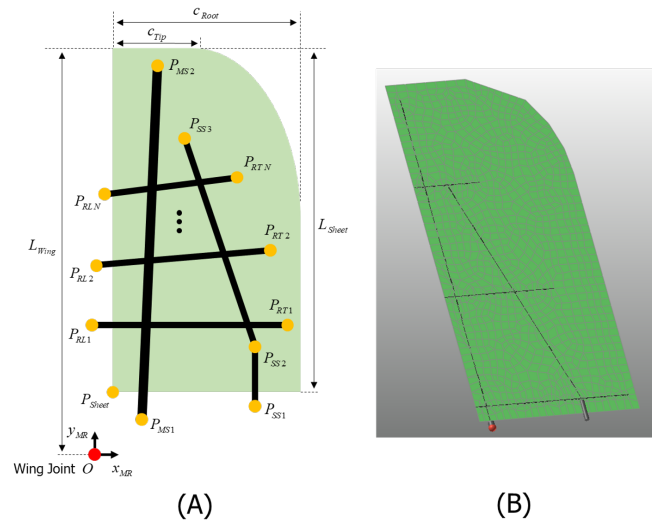


Figure 4: (A) Schematic diagram of the main wing, (B) Flexible wing model with structural mesh

ble model as a structural model, can provide highly efficient and reliable analysis results although its accuracy is lower than FSI using CFD and CSD solvers.

2.4 Wing Kinematics

The ornithopter model used in this study employs a flapping mechanism based on the planar crank-rocker mechanism, which is one of the widely used mechanisms in ornithopters. The schematic diagram of the planar crank-rocker mechanism is illustrated in Figure 5, where the angle of the fixed link is denoted as ϕ_0 , the length of the fixed link is denoted as L_0 , the length of the crank is denoted as L_1 , the length of the coupler is denoted as L_2 , the length of the rocker is denoted as L_3 , the transmission angle is denoted as μ , and the flapping angle is denoted as θ .

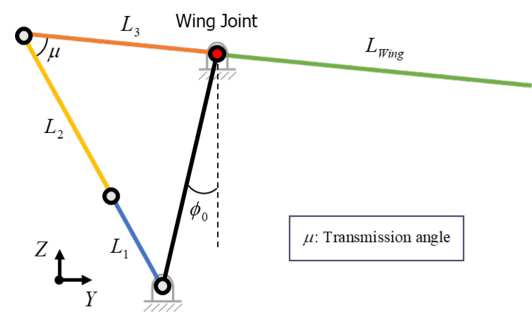


Figure 5: Schematic diagram of planar crank-rocker

The length of each link is obtained through parameter optimization. The optimization problem is solved to search for design parameters with an average transmission angle close to 90deg when given the minimum and maximum transmission angles, upside flapping angle, and downside flapping angle.

In this study, the minimum and maximum transmission angles are set to 45deg and 135deg, respectively. Also, the up-side and downside flapping angles are set to 45deg and 15deg, respectively. The optimization is performed using the genetic algorithm, one of the global optimization algorithms. The genetic algorithm generates a random population and keeps the elite child for the next generation. It reaches the global minimum by utilizing crossover and mutation algorithms. In this optimization problem, the population size is 5000 which is 1000 times the number of inputs (ϕ_0 , L_0 , L_1 , L_2 , L_3); elite-child and crossover rates are, respectively, 4% and 70% (mutation rate = 26%). The final outputs of the optimization are the angle of the fixed link and each length of the links, and the values are as follows: $\phi_0 = 12deg$, $L_0 = 37.6mm$, $L_1 = 10.0mm$, $L_2 = 33.5mm$, $L_3 = 20.0mm$).

3 RESULTS AND DISCUSSION

3.1 Validation for Aerodynamic Model

In order to validate the proposed aerodynamic model (modified UVLM with the PLEV model), wind tunnel tests were conducted on two different wing models: a rigid wing model and a flexible wing model. A rectangular plate wing was selected as the wing shape for the test. Both wings have 130mm of chord and 250mm of wing length: the aspect ratio is almost two. The rigid wing model consists of a balsa wood wing sheet with a thickness of 1 mm, and main- and subspars made of carbon fiber reinforced plastic (CFRP) pipes with outer and inner diameters of 4 mm and 3 mm, respectively. The flexible wing model has a wing sheet made of polyester with a thickness of 0.012 mm and a main spar made of a CFRP pipe with the same dimensions as the rigid wing model, but no sub-spar. The rib of the flexible wing model is made of a CFRP rod with a diameter of 0.8 mm.



Figure 6: Test model mounted in the KAIST subsonic wind tunnel

Wind tunnel tests were performed at a flow speed of 5.5 m/s and at different geometric angles of attack ranging from 0 to 25 degrees, with a flapping frequency of 2.5 Hz. The lift and thrust were measured using a 3-axis load cell (MC15-3C-50), while the flapping motion was actuated by a DC motor (GM35-2932E) and measured using a laser displacement

sensor (IL-600). The test models are mounted in the KAIST subsonic wind tunnel as shown in Figure 6.

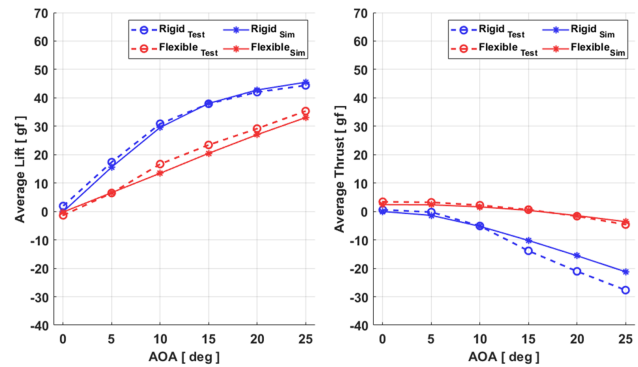


Figure 7: Cycle-averaged lift and thrust of test models

Figure 7 shows the cycle-averaged lift and thrust of the test models. The validation results of the proposed aerodynamic model using the modified UVLM showed a lift error of less than 1.7% and a thrust error of less than 13.5% for the rigid wing model. For the flexible wing model, the model showed a lift error of less than 5.1% and a thrust error of less than 16.9%. While there is room for improvement in terms of thrust estimation, it can be concluded that the modified UVLM provides a sufficiently accurate estimation for flapping aerodynamics in forward flight. The proposed approach takes only 42 seconds for the three cycles of the flexible wing model. Although the validation model is a single wing, the computation time within one minute means that the proposed FSI model is quite efficient in terms of computational time and can guarantee the fidelity of unsteady aerodynamic analysis for the flapping-wing model.

3.2 ORNithopter Integrated Simulation Program

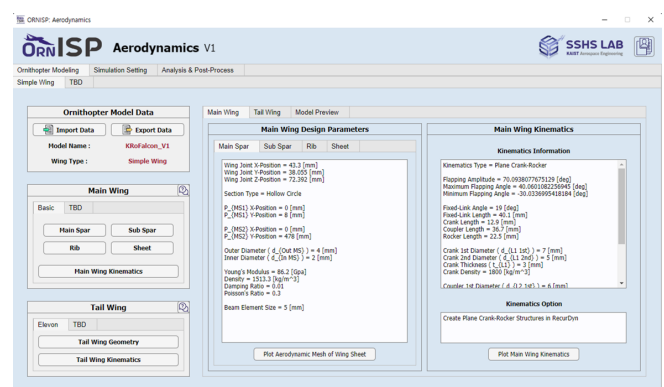


Figure 8: Main screen of ORNISP

ORNithopter Integrated Simulation Program (ORNISP) is an aimed program for aerodynamic analysis in designing ornithopters. It was developed as a MATLAB App, which

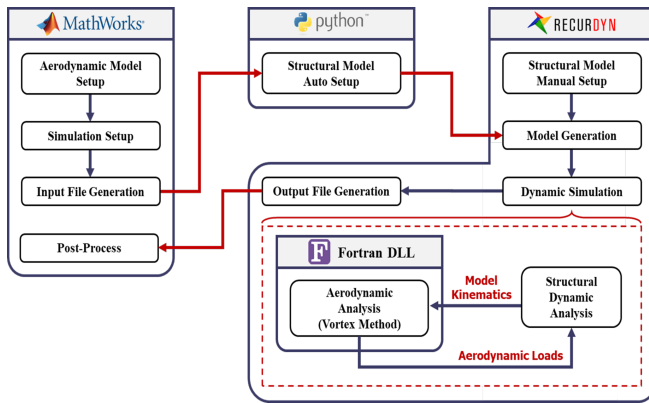


Figure 9: Configuration and flow diagram of ORNISP

consists of MATLAB, Python, and RecurDyn, and its main screen is shown in Figure 8. The program has a dedicated data format (*.omdat) for managing model data, and all information, such as the design parameters, kinematics, and UVLM simulation parameters of the ornithopter model, is classified and managed. Figure 9 shows the configuration and flow diagram of ORNISP. The MATLAB GUI allows adjusting the parameters of the wing flap mechanism model to be analyzed. Additionally, input files that have geometric information about the wing configuration are generated using MATLAB to create wing structures in RecurDyn. In this case, automatic model creation of the ornithopter simulation model using a Python script is possible.

The modified UVLM with the PLEV model is compiled as Fortran DLL and used in RecurDyn, enabling fluid-structure coupled analysis. After the analysis is finished, the exported CSV file can be used to perform the aerodynamic force computation and wake visualization. With ORNISP, it is possible to perform aerodynamic analysis for designing ornithopters without the extensive computational requirements of CFD and CSD solvers. The integration of the modified UVLM and PLEV models with RecurDyn, along with the flexibility of the wing model, allows for a comprehensive understanding of the aerodynamic characteristics of ornithopters during the initial design stages. Overall, ORNISP provides a user-friendly interface and automated workflow for the aerodynamic analysis of ornithopter designs.

3.3 Effect of Wing Flexibility on Aerodynamics

Unsteady aerodynamic simulations were performed for the two ornithopter models whose configurations are shown in Figure 1: rigid main wing and flexible main wing. Both wings have 200mm of mean chord and 490mm of wing length, and the overall geometric shape and dimensions are shown in Figure 10. The flight condition of the ornithopter model was determined based on the reduced frequency of Festo's Smart Bird[3]. Hence, in this study, the flapping frequency of 3.5 Hz, the forward flight speed of 6 m/s, and the

geometric angle of attack of 5 deg were selected.

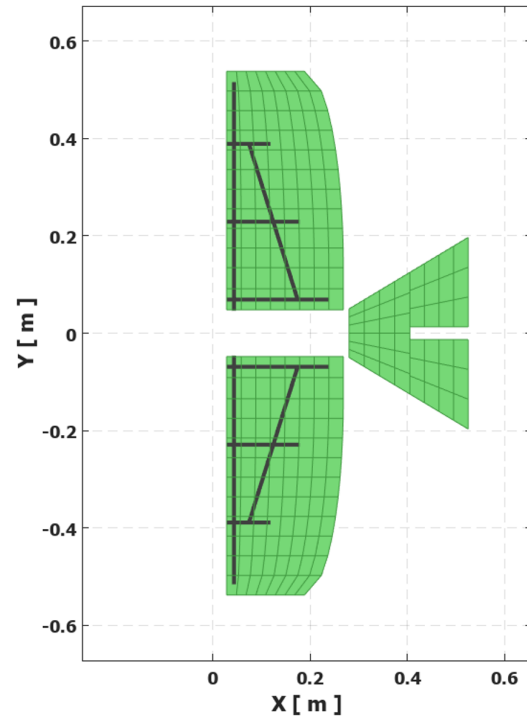


Figure 10: Geometric shape and dimensions of the simulation model

Figure 11 illustrates the comparison between the rigid wing and flexible wing models in terms of the lift coefficient (C_L). It can be observed that the rigid wing model produces a lift value similar to that of the flexible wing model. However, as shown in Figure 12, the rigid wing exhibits a negative mean value of thrust coefficient (C_T) under the given flight conditions, indicating its inability to achieve forward flight. In contrast, the flexible wing model demonstrates zero thrust under the given flight conditions, indicating that it is in a trim state of forward flight and capable of maintaining stable flight.

These results emphasize the importance of considering wing flexibility during the initial design stage of a bird-inspired ornithopter. The integration of flexible wing models, as demonstrated in ORNISP, allows for a more accurate assessment of the ornithopter's aerodynamic performance. By enabling the exploration of various wing designs and their effects on aerodynamics, ORNISP can provide valuable insights for optimizing the design of ornithopters with enhanced flight capabilities.

4 CONCLUSION

In this study, an integrated simulation program, named ORNithopter Integrated Simulation Program (ORNISP), has been proposed for the initial design of a bird-inspired ornithopter. To address the computational time limitations of

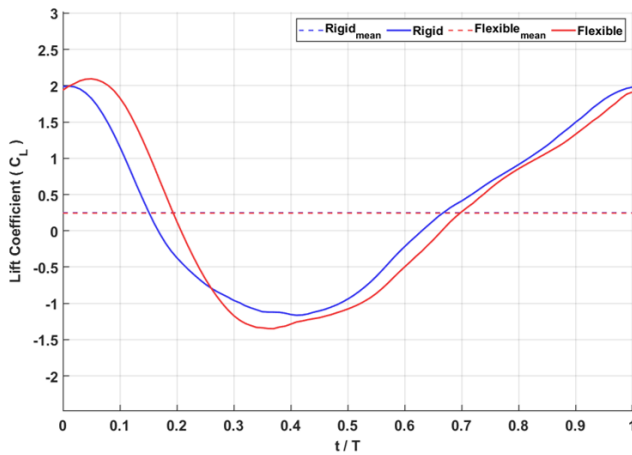


Figure 11: Time history of lift coefficient during one cycle

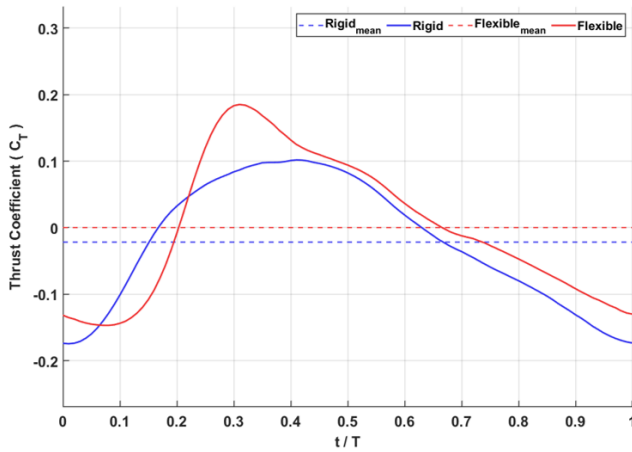


Figure 12: Time history of thrust coefficient during one cycle

CFD and CSD solvers, the UVLM was modified and implemented to efficiently compute the aerodynamic loads generated by the flapping motion. MFBF analysis was conducted using the commercial software RecurDyn. A flexible wing model, represented by beam and shell elements, was modeled and converted into a modal-based flexible model. The wing kinematics of the ornithopter model was based on the widely used planar crank-rocker mechanism. The proposed aerodynamic model was validated against experimental results, demonstrating its reliable accuracy (cycle-averaged lift error of less than 5.1%, thrust error of less than 16.9%) for the flexible wing model. The configuration of ORNISP was introduced. The effect of wing flexibility on ornithopter aerodynamics was analyzed. It was observed that considering wing flexibility is crucial, as demonstrated by the comparison between rigid and flexible wing models. The flexible wing model exhibited stable flight under the given forward flight condition, whereas the rigid wing model showed unfavorable

flight characteristics (negative cycle-averaged thrust).

In conclusion, this research presents ORNISP as a promising integrated simulation program for the efficient and effective initial design of bird-inspired ornithopters. ORNISP offers a user-friendly interface and automated workflow, eliminating the extensive computational requirements associated with CFD and CSD solvers. Future work will focus on expanding the capabilities of ORNISP to include more complex ornithopter models and wing kinematics, as well as broadening its application range to other bio-inspired micro air vehicles.

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 (2020M3C1C1A01083415).

REFERENCES

- [1] A. Chen, B. Song, Z. Wang, D. Xue, and K. Liu. A novel actuation strategy for an agile bioinspired fwav performing a morphing-coupled wingbeat pattern. *IEEE Transactions on Robotics*, 39(1):452–469, 2023.
- [2] G. A. Folkertsma, W. Straatman, N. Nijenhuis, C. H. Venner, and S. Stramigioli. Robird: A robotic bird of prey. *IEEE Robotics Automation Magazine*, 24(3):22–29, 2017.
- [3] W. Send, M. Fischer, K. Jebens, R. Mugrauer, A. Nagarathinam, and F. Scharstein. Artificial hinged-wing bird with active torsion and partially linear kinematics. In *28th ICAS Congress*, Brisbane, Australia, 2012.
- [4] J. Gerdes, A. Holness, A. Perez-Rosado, L. Roberts, A. Greisinger, E. Barnett, J. Kempny, D. Lingam, C.-H. Yeh, H. A. Bruck, and S. K. Gupta. Robo raven: A flapping-wing air vehicle with highly compliant and independently controlled wings. *Soft Robotics*, 1(4):275–288, 2014.
- [5] J. Tang, S. Chimakurthi, R. Palacios, C. E. S. Cesnik, and W. Shyy. Computational fluid-structure interaction of a deformable flapping wing for micro air vehicle applications. In *46th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, 2008.
- [6] T. Nakata and H. Liu. A fluid-structure interaction model of insect flight with flexible wings. *Journal of Computational Physics*, 231(4):1822–1847, 2012.
- [7] V. Joshi, R. K. Jaiman, and C. Ollivier-Gooch. A variational flexible multibody formulation for partitioned fluid-structure interaction: Application to bat-inspired drones and unmanned air-vehicles. *Computers*

and *Mathematics with Applications*, 80(12):2707–2737, 2020.

- [8] W. Yang, L. Wang, and B. Song. Dove: A biomimetic flapping-wing micro air vehicle. *International Journal of Micro Air Vehicles*, 10(1):70–84, 2018.
- [9] Q. Wang, J. F. L. Goosen, and F. van Keulen. An efficient fluid-structure interaction model for optimizing twistable flapping wings. *Journal of Fluids and Structures*, 73:82–99, 2017.
- [10] R. Schwab, E. Johnson, and M. Jankauski. A novel fluid–structure interaction framework for flapping, flexible wings. *Journal of Vibration and Acoustics*, 141(6):061002, 2019.
- [11] J. Wu and M. Sun. The influence of the wake of a flapping wing on the production of aerodynamic forces. *Acta Mech Sinica*, 21:411–418, 2005.
- [12] D.-K. Kim, J.-S. Lee, and J.-H. Han. Improved aerodynamic model for efficient analysis of flapping-wing flight. *AIAA Journal*, 49(4):868–872, 2011.
- [13] J. Katz and A. Plotkin. *Low-Speed Aerodynamics*. Cambridge University Press, Cambridge, UK, 2001.
- [14] S. A. Ansari, R. Żbikowski, and K. Knowles. Non-linear unsteady aerodynamic model for insect-like flapping wings in the hover. part 2: Implementation and validation. *Proc. Inst. Mech. Eng. Part G: Journal of Aerospace Engineering*, 220(3):169–186, 2006.
- [15] H.-H. Yang, S.-G. Lee, R. Addo-Akoto, and J.-H. Han. Parameter optimization of foldable flapping-wing mechanism for maximum lift. *Journal of Mechanisms and Robotics*, 16(3):031002, 2024.
- [16] A. T. Nguyen and J.-H. Han. Wing flexibility effects on the flight performance of an insect-like flapping-wing micro-air vehicle. *Aerospace Science and Technology*, 79:468–481, 2018.

3D Human Pose reconstruction Single-pixel imaging

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.

ABSTRACT

Three-dimensional (3D) human pose estimation is a fundamental task in computer vision and has important applications in fields such as gaming, sports analysis, and surveillance. Due to their high mobility and ability to capture images from different angles, the use of drones has rapidly increased in various applications in recent years. However, 3D human pose estimation from drone images remains a challenging task due to the complexity of the problem and the limited resolution of the images captured by the drone. Single pixel imaging (SPI) is a novel imaging technique that has recently gained attention in the field of computer vision due to its ability to capture high-resolution images with low-cost hardware. In SPI, a Hadamard pattern is projected onto the object of interest, and a single pixel detector measures the light that is reflected back from the object. In this paper, we propose a novel approach for 3D human pose estimation from drone images using SPI. The proposed method consists of three main steps: 1) SPI-based image acquisition, 2) feature extraction using deep learning, and 3) 3D pose estimation using a regression-based approach. To evaluate the proposed method, we collected a dataset of drone images of human subjects in different poses. The experimental results demonstrate that our approach achieves state-of-the-art performance in 3D human pose estimation from drone images. Compared to existing methods, our approach is computationally efficient and requires low-cost hardware.

1 INTRODUCTION

Three-dimensional (3D) pose estimation has become an essential tool in many fields, including robotics [1], augmented reality [2], unmanned Aerial Vehicles (UAVs)[3], and virtual reality[4], to name a few. It involves reconstructing an accurate 3D model of a human body from a single 2D image, which is challenging due to the inherent ill-posed nature of the problem [5]. However, advancements in computational techniques and algorithms have paved the way for exciting applications in various fields [6]. In this context, 3D pose esti-

mation has been used in drone technology to improve surveillance [7], emergency response [8], animal conservation [9], and infrastructure inspection [10].

Reconstructing an accurate human shape from imperfect input data, accounting for non-rigid deformations and joint articulations, is a challenging task. However, recent advances in deep learning techniques have made it possible to achieve end-to-end reconstruction of human shape [11]. The Skinned Multi-Person Linear model (SMPL-X) [12] offers a compact representation for 3D human shape, and has been integrated with deep neural networks for 3D human reconstruction from RGB images. This integration involves using deep neural networks to extract powerful image features, followed by direct regression of SMPL-X shape and pose parameters [13].

Different technologies, such as RGB cameras [13], thermal cameras [14], and IR-UWB RADAR [15], exist for estimating human pose. However, traditional cameras have limitations in low-light conditions, making them less useful for nighttime surveillance applications. In contrast, single-pixel camera (SPC) systems offer a promising solution to these limitations [16]. By exploiting the power of deep learning techniques, SPCs can reconstruct high-quality images from sparse measurements, making them an ideal candidate for detecting 3D human pose in nighttime surveillance applications. One key advantage of SPCs over traditional cameras is their ability to capture images in the near-infrared (NIR) spectrum, which provides better visibility in low-light conditions. Combining SPC technology with time-of-flight (TOF) sensing [17], we can obtain 2D/3D images of the environment, providing additional information about the location and movement of objects. The use of SPCs in surveillance applications is not limited to nighttime environments. They can also be used to capture images in harsh environments where traditional cameras may fail [18].

In this work, the authors propose a single-pixel imaging (SPI) vision system with active illumination in the NIR wavelength range of 850-1500 nm, which can be employed using single InGaAs photodetectors. As a detection strategy, they use a U^2 -net [19] to remove the background of the SPI image and identify the object for segmentation of the area of interest containing the element to detect. They then use Vision Transformers to perform silhouette analysis-based gait recognition for human identification [20]. The information is used to generate a 3D model through the VIBE method [21], which predicts SMPL-X body model parameters using a convolutional neural network pretrained on the AMASS dataset [22], for single-image body pose and shape estimation. This approach can improve the detection and surveillance capabilities

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

ties of drones, especially in low-light and harsh environments. Therefore, in this work, we propose the following:

- Investigating the potential of Single-Pixel Imaging in generating 3D human pose from low-resolution 2D images.
- This work focuses on a novel and difficult objective of forecasting the 3D hand pose using a single 2D binary mask acquired through NIR-SPI imaging.

2 SINGLE-PIXEL IMAGE RECONSTRUCTION

The technique known as single-pixel imaging (SPI) [17] reconstructs images by measuring correlated intensity on a detector without spatial resolution. To achieve this, SPI cameras use spatial light modulators (SLMs) like Digital Micro-Mirror Devices (DMD) to create structured light patterns using Hadamard pattern for scene interrogation [17]. There are two architectures in which SPI cameras operate, namely structured detection and structured illumination, as depicted in Figure 1.

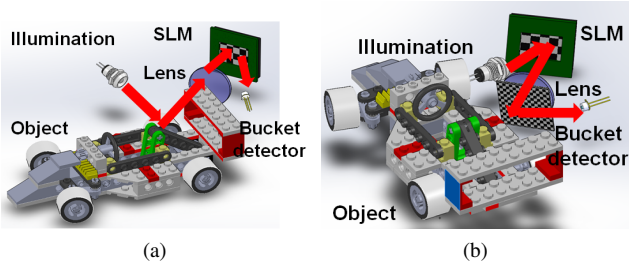


Figure 1: Two different approaches applied to SPI: (a) structured detection, and (b) structured illumination [17].

In structured detection, the object is illuminated with light, and the reflected light is projected onto an SLM, followed by detection using a bucket detector. On the other hand, in structured illumination, the light source is modulated by the SLM Φ_i and illuminates the object $O(x, y)$, and the reflected light is detected by a bucket detector and converted into an electrical signal S_i by Eq.(1) [17].

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

Here, α is a constant factor that depends on the optoelectronic response of the photodetector.

As the light's spatial pattern and the reflected light from the object correlate to an electrical signal, projecting a sequence of spatial patterns produces a sequence of electrical signals that can be used for computational image reconstruction. Therefore, the image $I(x, y)$ is reconstructed from the captured signal S_i and the corresponding pattern Φ_i using Eq.(2) [17].

$$I(x, y) = \alpha \sum_{x=1}^M \sum_{y=1}^N S_i \Phi_i(x, y) \quad (2)$$

To generate Hadamard-like patterns Φ_i using active illumination, this work uses an array of 32 x 32 NIR-LEDs that emit radiation with a peak wavelength of 1550 nm. This wavelength is chosen due to reduced scattering by water and reduced water absorption coefficients. The NIR-LED array is placed perpendicular to the lens's focal length to project the light pattern to infinity. However, given the array's size, the patterns are projected up to a distance of 0.3-3 meters. Although the active illumination approach does not fully illuminate the object, the technique of Fast Super Resolution CNN (FSRCNN) can reconstruct high-quality images [23]. The active illumination approach offers several advantages, including operating in different outdoor weather conditions, low-level illumination scenarios, and being less

2.1 Single-pixel camera (SPC)

This work introduces the concept of using structured illumination to enhance the quality of images captured in challenging lighting conditions, such as strong backlight and stray light. To achieve this, the structured illumination is provided by an array of 32 x 32 NIR-LEDs with a peak wavelength of 1550 nm, and a time-of-flight (TOF) system wavelength of 850 nm. The illumination is detected by an InGaAs photodiode (SPD). This active illumination approach has numerous advantages, including its ability to function under various outdoor weather conditions, low-level illumination, and reduced sensitivity to background radiation noise.

The proposed architecture for this Near-Infrared Single-pixel imaging (NIR-SPI) system is composed of two main parts (See Fig. 5). The first part includes the essential components for generating images, which are an InGaAs photodetector, an array of NIR-LEDs, a TOF system, and an ADC. The second part is responsible for processing the electrical output signal from the SPD module. It accomplishes this by digitizing the signal using the ADC and then using a Graphics Processing Unit (GPU) to process the data. The GPU unit used for this system is the Jetson Xavier NX, which generates Hadamard patterns and processes data from the ADC. It then runs the OMP-GPU Algorithm to generate 2D images [18].

3 HUMAN MODELING

Parametric human models like SMPL-X [12] offer a succinct representation of human shapes by using shape and pose parameters to encode variations [24]. The SMPL-X model offers several benefits, such as the disentanglement of human shape and pose, which allows for independent analysis and control of each [21]-[25]. It also avoids modeling rugged and twisted shapes directly, which can be problematic for neural network-based methods [26]-[27], by using a skinning process to model deformation. Moreover, SMPL-X is differentiable and can be easily integrated with neural networks [25].

Therefore, we have chosen to use SMPL-X as the underlying representation for modeling 3D humans in our research.

The SMPL-X model includes shape parameters β , pose parameters $\theta \in R^{3K}$, and global translation parameters. Body pose is defined by a skeleton rig with $K = 24$ joints, including the body root, as shown in Fig. 2 are used for shape blending and encode global shape information, while pose parameters are used for pose blending and skinning and encode local information between adjacent joints, except for the root joint's pose parameters, which denote the global rotation of the entire shape. Note that SMPL-X's pose parameters denote the relative rotation from a joint to its parent, which differs from 2D or 3D human pose estimation [28], where the pose refers to joint locations. With β and θ , we can obtain the 3D body mesh $M = f_{SMPL}(\beta, \theta)$, where $M \in R^{N \times 3}$ is a triangulated surface with $N=6890$. The 3D SMPL-X model locations of the body joints X can be predicted with the body mesh using a pre-trained mapping matrix $W \in R^{K \times N}$, such that $X \in R^{K \times 3} = WM$ [29]. To project the body joints from 3D to 2D, we use the perspective camera model. Assuming the camera parameters are $\delta \in R$

4 PROPOSED METHOD

The figure 6 illustrates the steps involved in obtaining a 3D human model using near-infrared single-pixel imaging (NIR-SPI). The process involves multiple computer vision techniques to reconstruct the 3D pose of a human from a single low-resolution image. Each step is explained in detail below:

- Capture a low-resolution image of the human using a single pixel (see Fig. 2). The image is then adjusted for contrast to extract the basic shape of the person without revealing any details. The background of the image is removed using the U^2 -net [19] method, a deep learning model that can accurately segment the foreground and background of an image. The resulting image is a silhouette of the person, which only shows the outline without any details of the surface or texture.
- Use the identified poses to regress the SMPL-X human pose (pose θ , shape β , and camera s, R, T) using the Video and Image-based human Body pose Estimation (VIBE) method, a deep learning model that can estimate the 3D pose of a human from a single image or video.
- Create a 3D reconstruction of the human pose using a tool such as SMPL-X, which can fit the estimated pose to a 3D body model (see Fig. 2).

By following these steps, a 3D human model can be obtained from a single low-resolution image using NIR-SPI technology and computer vision techniques.

5 EXPERIMENTAL RESULTS

The results of the experiment demonstrate that the proposed methods can be used to obtain a 3D human model from NIR-SPI imaging at a distance of 1 meter from the SPC camera, and under night-time illumination conditions, after calibrating the SPC camera based on [16]. The process involves multiple steps that utilize various deep learning models, including U^2 -net and VIBE, to extract and estimate the 3D pose of a human from a single low-resolution image. Figure 2 depicts the experimental setup and results. To assess the efficacy of the proposed method, the researchers performed experiments on several datasets, including Silhouette-based 3D Human Pose Estimation [20], and the Human Pose SMPL-X dataset, such as the AMASS dataset [22].

5.1 Discussion: Proposed method

To evaluate the proposed network architecture depicted in Figure 6 (Appendix A), we tested various approaches for reconstructing human positions at night-time from a distance of 1 meter using the SMPL-X model. Our tests took into account the limited field of view of the SPC camera, which spans $74^\circ \times 57^\circ$. We captured NIR-SPI images of human poses, including sitting, standing, bending, and lying. During our analysis, we observed limitations in hand and body positions relative to the reference image, particularly in the bending position due to the loss of information in the input NIR-SPI image caused by reflection effects and low resolution in the reconstructed NIR-SPI image. Nevertheless, our results showed that for the standing and sitting positions (as depicted in Figure 3), the 3D human reconstruction demonstrated superior accuracy in vertex and joint positions (as presented in Table 1).

Table 1: **The results are Mean vertex to-vertex (V2V) (in mm) and Mean Per-Joint Position Error (MPJPE) (in mm) [30] body for the different human position (Lying, Bending, Sitting, and Standing)**

Human Pose	V2V error ↓	MPJPE error ↓
Lying	57.29	53.2
Bending	49.86	40.19
Sitting	34.2	33.7
Standing	42	41

6 NIR-SPI 3D HUMAN POSE RECONSTRUCTION APPLIED TO UAVS AUTONOMOUS APPLICATION RESCUES

The NIR SPI 3D Human Pose reconstruction technology has the potential to revolutionize autonomous rescues carried out by unmanned aerial vehicles (UAVs) (see Fig. 4). Using Near-Infrared Spectroscopy Imaging (NIR SPI), the technology captures the reflective properties of the human body's surface in 3D. This technology can be especially useful in identifying the exact location of individuals who may be

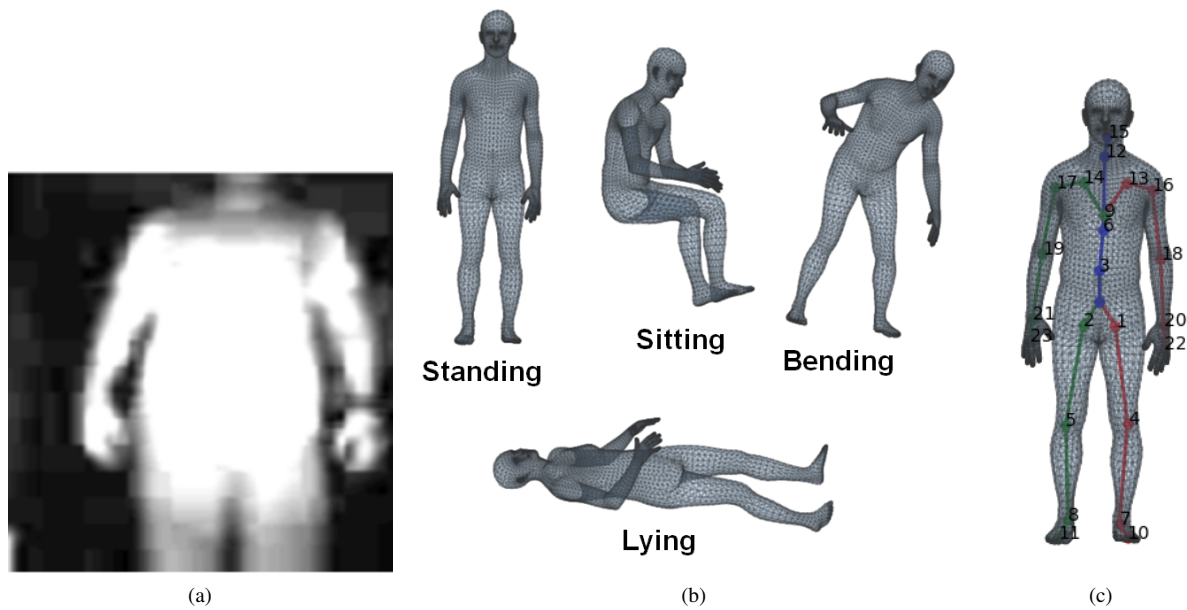


Figure 2: human poses but with same joint positions generate from NIR-SPI imaging: (a) Test NIR-SPI imaging, (b) SMPL-X model generate base on estimation pose (Standing, Sitting, Bending ,and Lying) , (c) SMPL-X model generate with joints

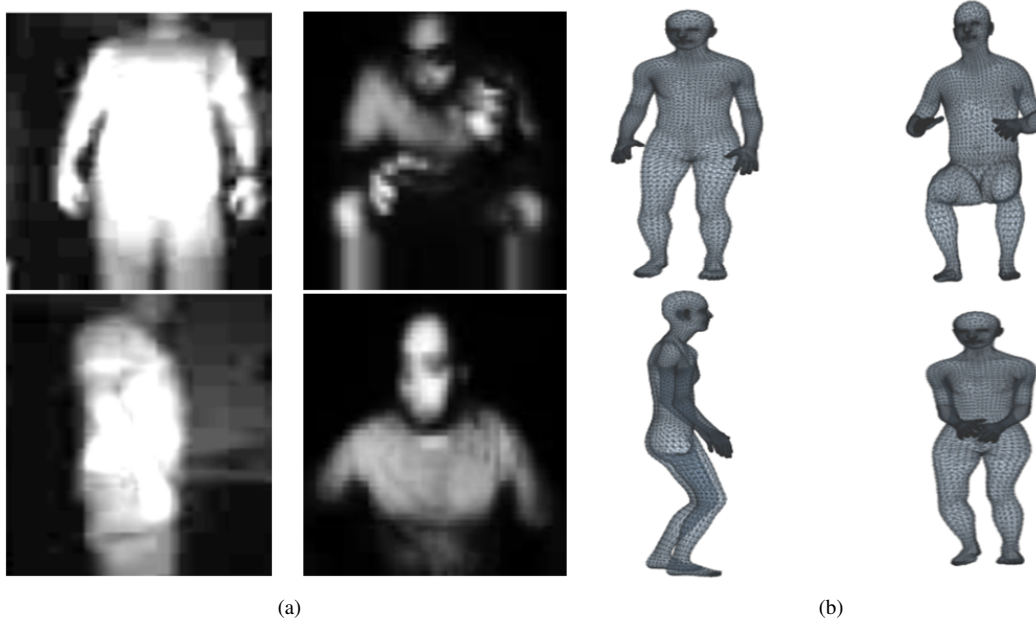


Figure 3: Capture human poses imaging at distance of 1 m: (a) Capture NIR-SPI imaging human pose standing, sitting, and bending, and (b) 3D human pose regression based on SMPL-X model.

trapped or injured in hazardous or hard-to-reach areas during UAV rescues. The 3D reconstruction of the human pose can provide invaluable information to rescue teams on the ground, allowing them to plan and execute a more effective rescue operation.

7 CONCLUSION

The study aimed to obtain a 3D model of the human body using NIR-SPI imaging for various poses, including standing, sitting, bending, and lying, with accuracy demonstrated in the V2V and MPJPE error table. Although the approach had limitations in hand positioning due to the low contrast of

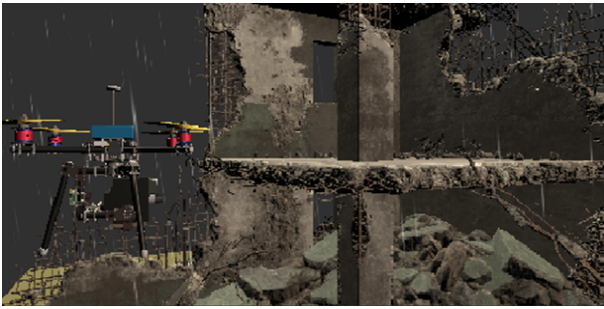


Figure 4: Schematic view to illustrate UAV (drone) environment hard-to-reach areas

the NIR-SPI image, the accurate estimation of the person's 3D pose through qualitative and quantitative evaluations of the level position of the core person detection was shown. These findings demonstrate the potential of the proposed approach for 3D human modeling from a single low-resolution image.

In contrast, the SMPL-X model presented in this study captures the body, face, and hands simultaneously and fits the model to a single NIR-SPI image and 2D joint detections. The study demonstrated that the SMPL-X model can capture bodies, hands, and faces from NIR-SPI images. However, the model had higher errors for bending and lying poses, indicating limitations in the pose parameters θ . Therefore, the study recommends implementing a compensation model in future applications. The authors also suggest future work involving the development of a dataset of SMPL-X fits in real-world scenarios and the direct regression of SMPL-X parameters from NIR-SPI images. Overall, this study is a crucial step towards capturing expressive body, hand, and face movements from an NIR-SPI image, with potential applications in the fields of intelligent automation, unmanned aerial vehicles, and autonomous vehicles.

ACKNOWLEDGEMENTS

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

REFERENCES

- [1] Christian Zimmermann, Tim Welschhold, Christian Dornhege, Wolfram Burgard, and Thomas Brox. 3d human pose estimation in rgbd images for robotic task learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1986–1992, 2018.
- [2] P. Sudhaman, P. Surendiren, Sherif.S Meeran, and P.C. Kishore Raja. Augmented reality in automation using virtual 3d models. In *2012 Third International Conference on Computing, Communication and Networking Technologies (ICCCNT'12)*, pages 1–4, 2012.
- [3] Nitin Saini, Elia Bonetto, Eric Price, Aamir Ahmad, and Michael J. Black. Airpose: Multi-view fusion network for aerial 3d human pose and shape estimation. *IEEE Robotics and Automation Letters*, 7:4805–4812, 2022.
- [4] Debangana Ram, Bholan3D VRath Roy, and Vaibhav Soni. A review on virtual reality for 3d virtual trial room. In *2022 IEEE World Conference on Applied Intelligence and Computing (AIC)*, pages 247–251, 2022.
- [5] Yueh-Ling Lin and Mao-Jiun J. Wang. Constructing 3d human model from 2d images. In *2010 IEEE 17th International Conference on Industrial Engineering and Engineering Management*, pages 1902–1906, 2010.
- [6] Wenming Meng, Tao Hu, and Shuai Li. 3d human pose estimation with adversarial learning. In *2019 International Conference on Virtual Reality and Visualization (ICVRV)*, pages 93–99, 2019.
- [7] Yap Wooi Hen and Raveendran Paramesran. Single camera 3d human pose estimation: A review of current techniques. In *2009 International Conference for Technical Postgraduates (TECHPOS)*, pages 1–8, 2009.
- [8] Savvas Papaioannou, Panayiotis Kolios, Theocharis Theocharides, Christos G. Panayiotou, and Marios M. Polycarpou. Towards automated 3d search planning for emergency response missions. *Journal of Intelligent & Robotic Systems*, 103(1):2, Aug 2021.
- [9] C.A. Johnson, J. Seidel, R.E. Carson, W.R. Gandler, A. Sofer, M.V. Green, and M.E. Daube-Witherspoon. Evaluation of 3d reconstruction algorithms for a small animal pet camera. In *1996 IEEE Nuclear Science Symposium. Conference Record*, volume 3, pages 1481–1485 vol.3, 1996.
- [10] Christos Papachristos, Kostas Alexis, Luis Rodolfo Garcia Carrillo, and Anthony Tzes. Distributed infrastructure inspection path planning for aerial robotics subject to time constraints. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 406–412, 2016.
- [11] Margarita Khokhlova, Cyrille Migniot, and Albert Dipanda. 3d visual-based human motion descriptors: A review. In *2016 12th International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*, pages 564–572, 2016.
- [12] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

- [13] Dongyue Chen, Yuanyuan Song, Fangzheng Liang, Teng Ma, Xiaoming Zhu, and Tong Jia. 3d human body reconstruction based on smpl model. *The Visual Computer*, Apr 2022.
- [14] Henry M. Clever, Zackory Erickson, Ariel Kapusta, Greg Turk, C. Karen Liu, and Charles C. Kemp. Bodies at rest: 3d human pose and shape estimation from a pressure image using synthetic data. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6214–6223, 2020.
- [15] Gon Woo Kim, Sang Won Lee, Ha Young Son, and Kae Won Choi. A study on 3d human pose estimation using through-wall ir-uwv radar and transformer. *IEEE Access*, 11:15082–15095, 2023.
- [16] Carlos Alexander Osorio Quero, Daniel Durini, Jose de Jesus Rangel-Magdaleno, Jose Martinez-Carranza, and Ruben Ramos-Garcia. 2d nir-spi spatial resolution evaluation under scattering condition. In *2022 19th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, pages 1–6, 2022.
- [17] 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.
- [18] 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.
- [19] Xuebin Qin, Zichen Zhang, Chenyang Huang, Masood Dehghan, Osmar R. Zaiane, and Martin Jagersand. U2-net: Going deeper with nested u-structure for salient object detection. *Pattern Recognition*, 106:107404, 2020.
- [20] Ryosuke Hori. Silhouette-based 3d human pose estimation using a single wrist-mounted 360° camera, 2022.
- [21] Muhammed Kocabas, Nikos Athanasiou, and Michael J. Black. Vibe: Video inference for human body pose and shape estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [22] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of motion capture as surface shapes. In *International Conference on Computer Vision*, pages 5442–5451, October 2019.
- [23] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 391–407, Cham, 2016. Springer International Publishing.
- [24] Chun-Hao P. Huang, Hongwei Yi, Markus Höschle, Matvey Sifroshkin, Tsvetelina Alexiadis, Senya Polikovsky, Daniel Scharstein, and Michael J. Black. Capturing and inferring dense full-body human-scene contact. In *Proceedings IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 13274–13285, June 2022.
- [25] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [26] Or Litany, Alex Bronstein, Michael Bronstein, and Ameesh Makadia. Deformable shape completion with graph convolutional autoencoders. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1886–1895, 2018.
- [27] Gül Varol, Duygu Ceylan, Bryan Russell, Jimei Yang, Ersin Yumer, Ivan Laptev, and Cordelia Schmid. BodyNet: Volumetric inference of 3D human body shapes. In *ECCV*, 2018.
- [28] Helge Rhodin, Mathieu Salzmann, and Pascal Fua. Unsupervised geometry-aware representation for 3d human pose estimation. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 765–782, Cham, 2018. Springer International Publishing.
- [29] Xiangyu Xu, Hao Chen, Francesc Moreno-Noguer, László A. Jeni, and Fernando De la Torre. 3d human shape and pose from a single low-resolution image with self-supervised learning. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 284–300, Cham, 2020. Springer International Publishing.
- [30] Gyeongsik Moon, Juyong Chang, and Kyoung Mu Lee. V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

APPENDIX A: NIR-SPI SYSTEM VISION AND MODEL PROPOSED

NIR-SPI architecture comprises modulate in the figure 5 show an overall block diagram. The model proposed based in the feature extraction the NIR-SPI image (background remove and generation silhouette pose), the 3D pose estimation using a regression-based approach using the information the pose (shape parameters β and pose parameters θ) see Figure 6.

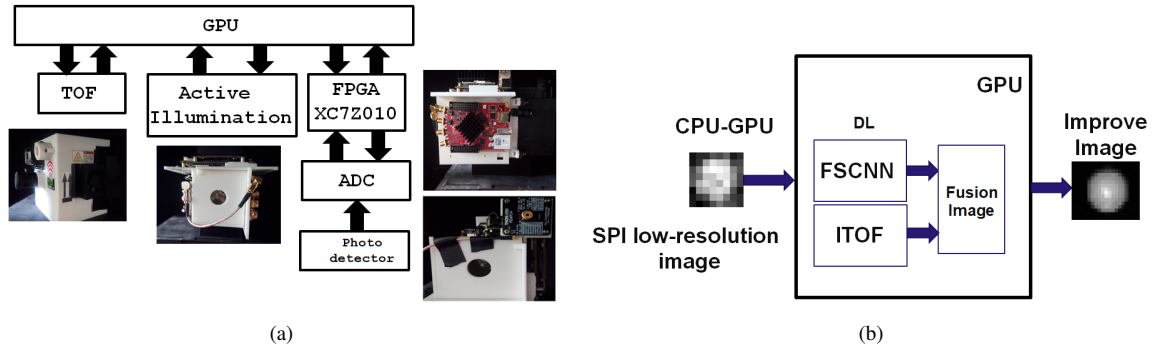


Figure 5: Overall block diagram: (a) Proposed vision system dimension 11 x 11 x 14 cm, focal length 20 cm, weight 1.2 kg, power consumption 45 W, first stage module photodiode, active illumination source, photodetector diode InGaAs FGA015, TOF system, second stage GPU unit and ADC, (b) diagram of the processing algorithm used by the proposed NIR-SPI vision system from low-resolution SPI image applying FSCNN network and fusion image with information captured way TOF.

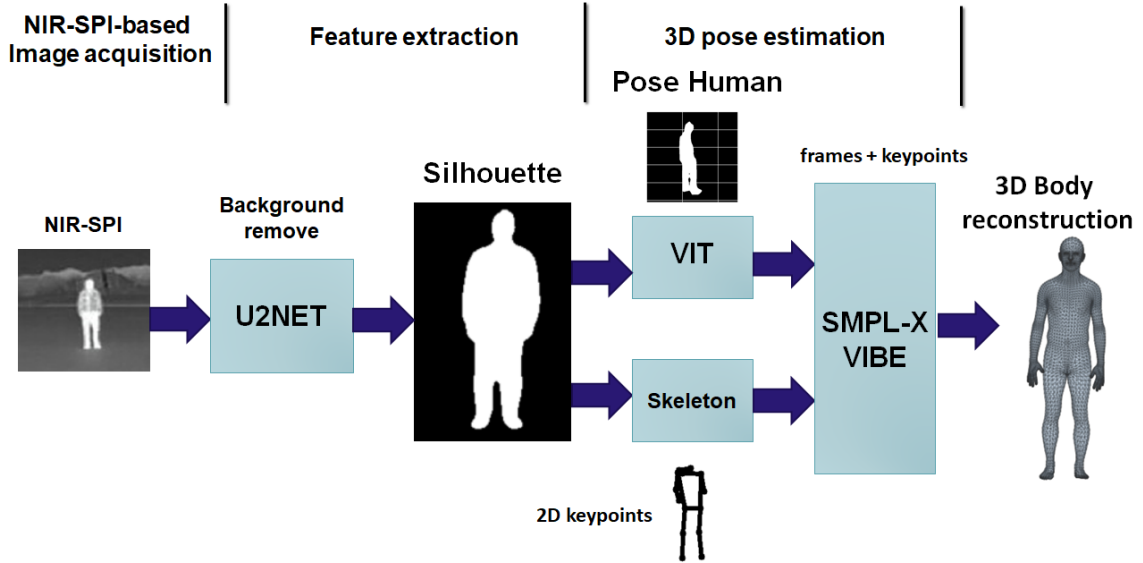


Figure 6: Overview of the proposed network architecture, which takes NIR single-pixel imaging input and outputs 3D body reconstruction based on SMPL-X shape and pose parameters. The entire network consists of three main modules: (i) NIR-SPI-based image acquisition, (ii) Feature extraction using deep learning : To extract the background, the NIR-SPI image is used to obtain the silhouette, (iii) 3D pose estimation using a regression-based approach: The silhouette image is used to obtain the GAIT features (shape estimation), which are then used to pose the human using Vision Transformers (VIT) and skeleton joint features. These features are used to pre-define the pose SMPL-X model. From the pre-defined parameters (pose θ , shape β and camera s, R, T), the SMPL-X model is fed to the off-the-shelf SMPL-X model to obtain the reconstructed 3D human mesh.

Hierarchical Continual Learning for Single Image Aerial Localisation

Aldrich A. Cabrera-Ponce, Manuel Martin-Ortiz, and Jose Martinez-Carranza*

Benemerita Universidad Autonoma de Puebla (BUAP), Puebla, Mexico

*Instituto Nacional de Astrofisica, Optica y Electronica (INAOE), Puebla, Mexico

ABSTRACT

Due to weather-related signal loss, GPS-based aerial localisation is challenging for Unmanned Aerial Vehicles (UAVs). Consequently, vision-based techniques have been devised to tackle this problem by leveraging the cameras integrated into UAVs. The main objective is to achieve UAV localisation throughout a flight mission using aerial images and Convolutional Neural Networks (CNNs). To address this, we introduce an aerial localisation methodology that integrates continual learning and a multi-model approach, augmented by the concept of sub-mapping inspired by Simultaneous Localisation and Mapping (SLAM) systems. This methodology involves mapping an area into different zones and re-localising the camera when it reaches a known point. We compared results using the ORB-SLAM2, a keyframe searching method based on colour histograms, and a single model to validate our methodology. Our results demonstrate that it is possible to find corresponding sub-maps and acquire the camera pose from aerial images, achieving an average accuracy of 0.77 and a processing speed of 69 fps.

1 INTRODUCTION

Aerial localisation presents a challenge for UAVs that rely on GPS coordinates for outdoor flight missions. This reliance on GPS devices often hinders pose capture and has led to the development of several vision-based methods such as feature matching [1], Visual Odometry (VO) [2], and SLAM [3]. However, these methods are computationally expensive, leading to deep-learning methods with CNNs to estimate camera poses. For the latter, CNN architecture is trained using datasets with pose labels, resulting in a learning model with sufficient accuracy to estimate the camera poses from a single image.

Current research uses the PoseNet architecture [4] to regress the camera pose using a single image. This architecture can solve the aerial localisation problem and deter-

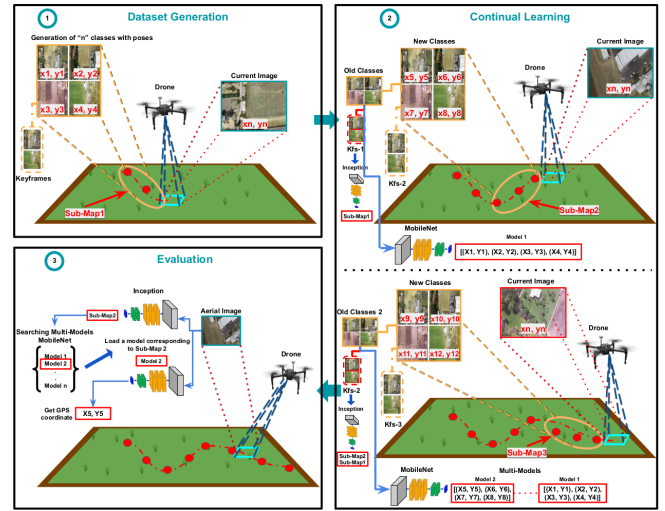


Figure 1: The outlined framework consists of three stages: 1) Dataset creation with poses designated as classes; 2) Continual learning using MobileNet for camera localisation and InceptionV4 to train the keyframes for sub-maps; 3) Multi-model evaluation, accomplished through keyframe searching from InceptionV4 results.

mine the camera poses on the UAV [5]. Nevertheless, training the model can be time-consuming during flight missions due to the large dataset and limited computational memory. An alternative approach to tackle this issue is to use continual learning strategies, which enable incremental training of CNN with small dataset samples and avoid catastrophic forgetting when new information arrives.

A latent replay strategy can be helpful in training models for in-flight missions, especially when dealing with constrained resources or uncertain surroundings. CNNs can adjust and enhance their performance without re-training, continuously updating the model with new information whilst leveraging previous knowledge. This consists of the repeatability of the last data patterns in external memory and integrating them with the new incoming data. Furthermore, the segmentation of the environment into distinct sections can yield advantages, facilitating a more concentrated and targeted approach to the learning process. Overall, continual

*Email address(es): carranza@inaoe.mx

learning strategies can improve the performance of models used, where real-time adaptability is necessary for localisation tasks.

Motivated by the above, we propose a methodology that learns hierarchically with two networks employing a latent replay strategy. To this end, we generated the sub-maps throughout the trajectory, where representative keyframes were extracted for each sub-map and trained the first network to localise the map in the area. Then, we train the second network with camera pose information in a multi-model fashion where each model generated represents one sub-map with flight coordinates information. Finally, we evaluate a testing dataset to identify the sub-map and load the corresponding model to obtain camera poses. An overview of our framework, which is divided into three stages, is presented in Figure 1.

To present our work, this paper is organised as follows. Section 2 discusses related works on localisation and continual learning approaches. Section 3 outlines the dataset generation process, sub-maps creation, and continual learning using the two architectures. Section 4 presents the experimental design and a comparison with other methods for localisation. Finally, conclusions and future work are summarised in Section 5.

2 RELATED WORK

Aerial localisation for outdoor scenarios is often challenging for UAV flight missions, especially for autonomous navigation tasks. Several works have proposed solutions to obtain aerial localisation from methods such as multi-sensor fusion with vision [6], visual odometry [7], feature matching [1], and SLAM. The latter has played an essential role in robotics, allowing us to map scenarios from features extracted in images. Therefore, one of the best-known SLAM systems is ORB-SLAM2 [3, 8] used in inspection tasks where GPS is denied [9].

In contrast, deep learning has had a significant advance using convolutional neural networks (CNNs) for place recognition tasks [10], geo-localisation with cross-view and satellite imagery [11], and goal localisation [12]. The latter uses reinforcement learning to decouple scans on different goals finding nearby localisations. However, repetitive reproducibility may delay the acquisition of the pose. Instead, other methods obtain localisation by comparing map mosaics [13], dense scenario match [14], and using multi-task networks with visual odometry, and auxiliary learning [15].

In this way, PoseNet is an architecture designed for pose estimation from an image [4]. This advantage has been used for pose estimation, and localisation under multiple scenes, sharing the images' features across the entire scenario [16]. Nonetheless, pose acquisition is still linked to end-to-end training with large datasets. An alternative is to freeze the learning in some layers inside the architecture, reducing resource consumption in the training stage [17]. On the other

hand, in [5, 18], eliminate some layers in the CNN to leverage the time and parameterisation reduction with a compact network.

Due to the time and training constraints present in traditional learning, recent works have focused on continual learning strategies using small data [19, 20]. Strategies based on latent replay of data represent better performance in robotics tasks, keeping accuracy and avoiding catastrophic forgetting of prior learning [21–23]. However, these works are focused on continual learning for classification tasks, resulting in other solutions for localisation. For example, in [24, 25], they present hierarchical models for predicting semantic 2D worlds and incremental mapping from 3D measurements with stored representative features. These works allow prediction and scenario reconstruction to obtain the localisation using a single image.

On the other hand, [26] presents a method based on a buffer with 3D information of the scene and strategic samples. iMAP [27], maps an implicit scenario updating the discrepancies with keyframes to fill unseen parts. This has led to the goal of obtaining the pose using mapping and 3D modelling of the scenario, acquiring the localisation from a single image. To the same end, [28, 29] present methodologies for a continual SLAM using two architectures to learn and adapt in new scenarios and using a continual dictionary with a Quadratic Bayesian Surprise (QBS). Consequently, [30] present a sub-mapping approach with a multi-model process and continuous learning for aerial localisation.

Finally, these works give us a panorama where localisation is performed from sub-maps, multi-scenes, and multi-models using a hierarchical scheme. Motivated by the latter, in this paper, we propose continual hierarchical learning and adopt the SLAM concept to learn small parts of the scenario as a way of sub-mapping, thus creating learning models for each sub-map. Regarding SLAM, our approach can be seen as a hierarchical system that identifies keyframes to load a model corresponding to the sub-map and obtain the image's pose.

3 METHODOLOGY

This section outlines the methodology for continual hierarchical learning with two deep network architectures: MobileNetV1 and InceptionV4. Firstly, we collected aerial images from a monocular camera onboard the UAV and associated them with GPS coordinates during the flight. Afterwards, we save a keyframe set for each trajectory section, which we consider sub-maps of the whole path. Finally, we incrementally trained the networks to create a model containing the pose information and another model containing the submap information.

3.1 Dataset Generation

We split the dataset generation process into two steps. The first step involved obtaining aerial images paired with GPS coordinates, while the second entailed creating

keyframes to represent sub-maps. For the latter, the Robot Operating System (ROS) facilitated communication with the drone, and the Ground Control Station (GCS) received the image stream and corresponding GPS data. We perform initial flight missions to capture images at a resolution of 128×128 pixels and convert GPS coordinates into meters. Thus, we generate small samples of flight coordinates divided into classes, yielding about 50 classes for each trajectory. In addition, we defined sub-maps containing the information of 5 flight coordinates and created a new map when the drone covers a distance of 50 meters. Finally, we save keyframes for each submap to train the InceptionV4 network, whose information determines the trajectory zone to which they belong.

Figure 2 illustrates the creation and division of sub-maps with 5 flight coordinates and the storage keyframes with three images for each sub-map. These keyframes represent the images of the sub-maps beginning, middle and end, corresponding to classes 1, 3, and 5, respectively. Besides, we consider the distance between the points to create a new sub-map when the drone travels approximately 50 to 100 metres.

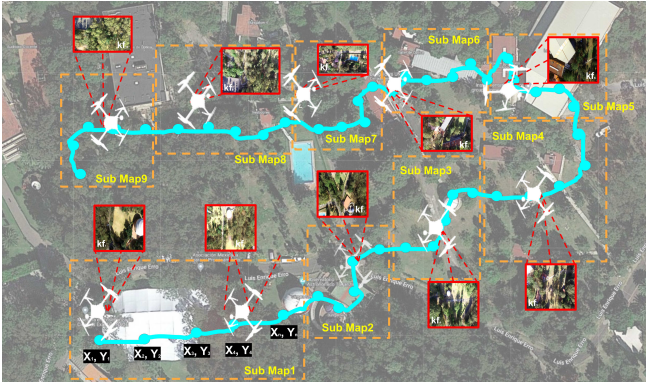


Figure 2: Training dataset acquisition, sub-maps creation, and keyframe saving. Cyan shows the flight drone trajectory, where the circles represent GPS coordinates. The orange rectangles illustrate the sub-maps created.

In Figure 3, we visually represent the trajectories overlaid in Google Earth using GPS-derived information. Simultaneously, Figure 4 illustrates the covered area of the routes transformed into meters. These trajectories represent the flights carried out to collect the training dataset and generate the keyframes, and the flight coordinates are presented by red circles in Figure 4. The length of the traversed trajectory is 0.53 km for trajectory 1, 1.4 km for the second, and 2.4 and 2.9 km for trajectories 3 and 4, respectively. Moreover, we have 47 flight coordinates for trajectory 1, 50 for the second and third trajectories, and 52 for the last trajectory, representing the classes along the entire path.

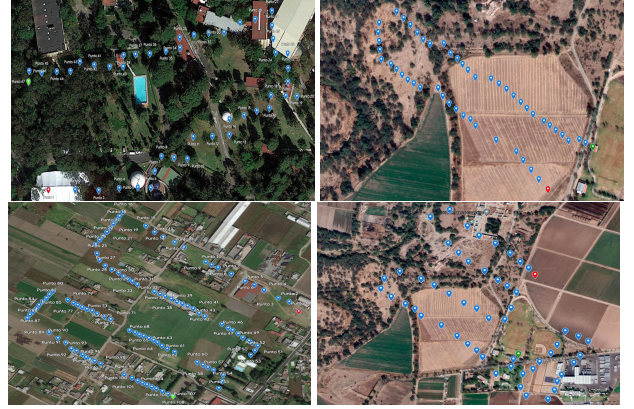


Figure 3: Flight paths into Google Earth with GPS information like waypoints. We show the beginning path in a red waypoint and the end in green.

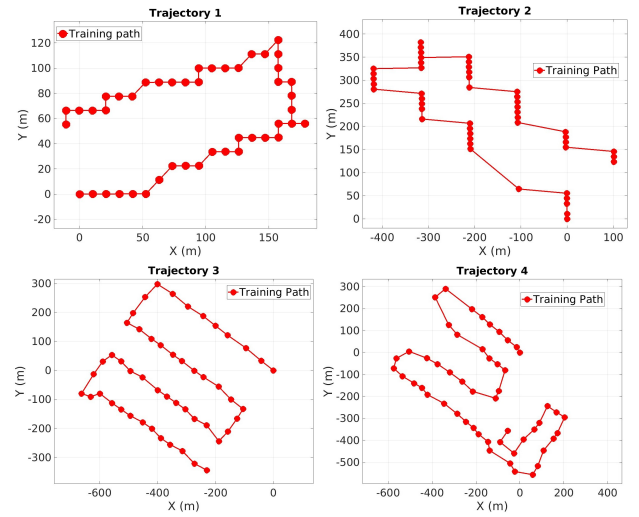


Figure 4: GPS coordinates in metres with a total trajectory length of 1) 0.53 km with 42 poses; 2) 1.4 km with 50 poses; 3) 2.4 km with 50 poses; 4) 2.9 km with 52 poses.

3.2 Continual Learning

For this section, we adopt the concept of sub-mapping used in SLAM systems, which divides the whole trajectory into several parts to achieve better data processing and establish aerial localisation under a hierarchical learning methodology. The goal is to acquire the image's poses once it is located within one of the sub-maps along the trajectory. To achieve this, we split the methodology into two parts and continually train two CNNs: MobileNetV1 and InceptionV4.

First, we use the continual learning strategy called latent replay to train MobileNetV1. This allows us to train on the fly information, saving the essential patterns in external memory, where each pattern set represents the classes. This method combines new data with old ones, rejuvenating the previously

learned weights and consolidating learning. As a result, the network can learn 1000 classes whose previous information is stored in a latent layer within an external memory in the *pool6* layer. However, we set a limit of continuous learning with around 50 classes to avoid saturating the memory with parameters and to prevent catastrophic forgetting.

In this way, the network continuously learns the 50 poses it receives from the information obtained from the drone, collecting 200 images for each established coordinate. Splitting the labels into subsections can improve the evaluation step, training a model with five coordinates representing one sub-map in the entire path. To train the InceptionV4, we save three keyframes of the sub-maps generated, as shown in Figure 5. The flow of information is trained with MobileNetV1, and the keyframes are stored for training the InceptionV4 network. Finally, we keep 27 keyframes for trajectory 1, 30 for trajectories 2 and 3, and 52 for trajectory 4.

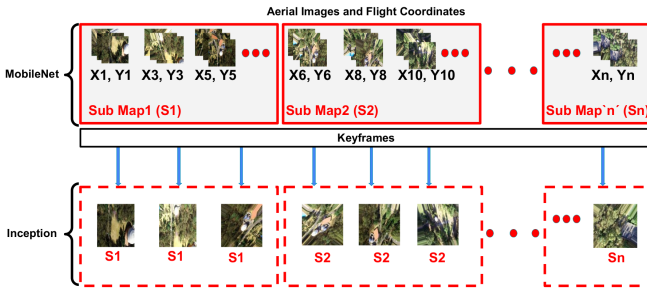


Figure 5: Continual training of the networks in a hierarchical way where MobileNet is trained using aerial images with information on flight coordinates and InceptionV4 with keyframes generated for each sub-map as labels.

Secondly, we employ continual learning of InceptionV4, which uses the keyframes and their labels to select the sub-map for the aerial image. The network is structured to learn an input image, whereby we encapsulate the features within a vector format. As a new keyframe of the matching class appears, we update the weights within a temporal vector and combine them with the previous ones. Conversely, if a different keyframe class arrive, we assign new features in a new index, expanding the features vector to adjust information from both classes. At the end of the training, we concatenate the features vector with the temporal vector, merging all keyframe features into one. This approach enables the network to learn from the continuous data stream and update its weights dynamically, improving its performance over time.

Thereby, hierarchically, we have a methodology capable of localising an aerial image inside the trajectory and acquiring the position. To carry out the continual learning of the networks, we take the following parameters summarised in Table 1. It is all on one computer with Cuda 11.1, PyTorch 1.9, and 8GB RAM with a GeForce 640M Nvidia card.

Table 1: Parameters used for continual training of the networks MobileNetV1 and InceptionV4.

Network	MobileNet	InceptionV4
Optimiser	SGD	SGD
Batch size	128	1
Epoch	20	1
Loss Function	Cross Entropy	Cross Entropy
Learning Rate	0.001	0.001
Latent layer	Pool6 Layer	Temporal Vector
Patterns/Features	3500	1536

4 EXPERIMENTS AND RESULTS

We carried out two experiments to assess the efficiency of our hierarchical aerial localisation approach. The first experiment aimed to identify the corresponding sub-map for the given aerial image using the InceptionV4 and a colour histogram-based method. In the second experiment, we leveraged the hierarchical framework to obtain the poses of an input image and ascertain the associated sub-map labels. We then loaded the MobileNet model with pose information to establish localisation. We also evaluated a single model, the ORB-SLAM2 re-localisation module, and a keyframe search using the colour histogram to compare our approach. Finally, we measured each method's performance speed expressed in frames per second (fps), running on Ubuntu 20.04, Python 3.8, OpenCV 3 and ROS Noetic.

4.1 Sub-Maps Results

In this experiment, we trained the InceptionV4 network on the fly using previously saved keyframes generated while creating sub-maps. Each keyframe is labelled with an index representing the corresponding sub-map number. When a new keyframe of a different class arrives, the network extracts the features and stores their weights in a temporary vector, merged with the previous ones. In this way, once the final sub-map is trained, the feature vector contains information on all keyframes learned, along with their respective labels. To evaluate the learning model, we conducted 4 test flights near the training trajectory, where each image was fed as the input to the network. The network's output provides the sub-map index and the total number of features corresponding to the closest keyframe. Thus, the network returned the label with the highest similarity of features found in the stored keyframes, which corresponded to the sub-map class.

We have implemented a keyframe search engine based on the image's colour to provide a comparison. Firstly, we divide the keyframe into five quadrants and extract the colour histogram, which is then concatenated into a single histogram. This provides more precision in finding descriptors during the evaluation step. Subsequently, we calculate the colour histogram of each image and compare it with the keyframes using the chi-square metric, where a value close to zero indicates a higher similarity with the keyframe. Tables 2 and

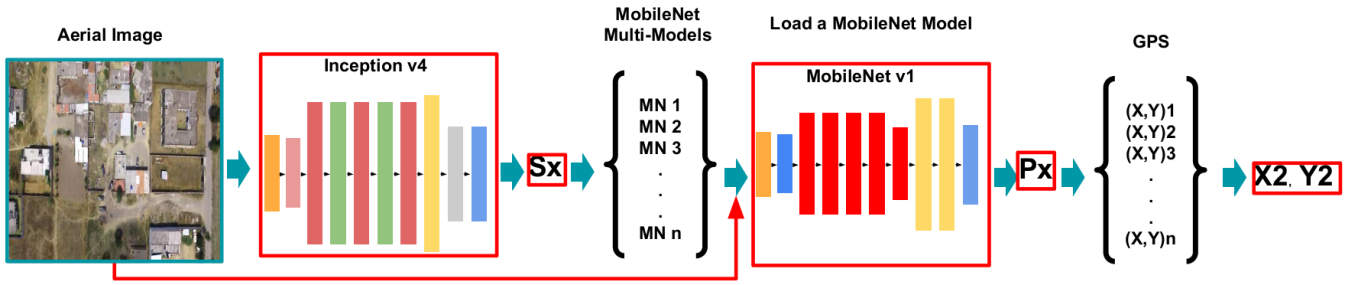


Figure 6: Hierarchical aerial localisation. We hierarchically evaluate each image, first passing for the InceptionV4 network to find the corresponding sub-map. Then, given the sub-map, we load the MobileNet model with information on 5 flight coordinates. Finally, the same image passes through the model to find the camera pose and get the aerial localisation using a single image.

Table 2: Accuracy results with the test dataset using a colour histogram descriptor to find the submap.

Trajectory	SubMap	Kfs Found	Accuracy
1	9	87	0.6041
2	10	65	0.5555
3	10	56	0.6666
4	10	310	0.5107

Table 3: Accuracy results with the test dataset using the InceptionV4 network to find the submap.

Trajectory	SubMaps	Kfs Found	Accuracy
1	9	117	0.8125
2	10	84	0.7179
3	10	60	0.7142
4	10	467	0.7693

3 present the result of the first experiment involving InceptionV4 and colour histogram. We also report the number of correctly localised images, their corresponding keyframes, and accuracy results. This experiment provides an overview of the keyframe search, highlighting the effectiveness of the InceptionV4 network in learning trajectory zones, given its continual training.

4.2 Hierarchical Learning Results

The second experiment evaluates our approach using the hierarchical concept for aerial localisation. For this, we follow the diagram in Figure 6, where we pass an input image through the InceptionV4 network, resulting in the sub-map label to which it corresponds. Then, we load the MobileNet model of the sub-map found and evaluate the input image, giving in the output one of the five classes learned. We argue that methodology can get the zone localisation and image's pose in a level fashion in contrast to the traditional localisa-

tion frameworks.

For comparison, we evaluate other methods to visualise aerial localisation using a single learning model, the ORB-SLAM2 localisation module, and a keyframe search engine using the colour histogram. The learning of a single model consists of continual learning with the latent replay strategy of the data. However, instead of having one model for each sub-map, we learn and update all classes in a single model. This can impact learning, leading to catastrophic forgetting of previous data when more information arrives on the network.

On the one hand, in the ORB-SLAM2 re-localisation module, we create the map using the training dataset and save the image poses in a text file. Then, we use the test dataset, deactivating the mapping to re-localise the images with feature matching. Thus, a test image with the same descriptors is automatically re-localised inside the map. Nevertheless, we take the distance between the test frame and the keyframe to recover the position by determining the coordinate corresponding to the nearby keyframe. On the other hand, we apply our same hierarchical methodology for the histogram results, changing the Inception network for the colour histogram, and we evaluate the test dataset. For this case, we define the bins of Hue, Saturation and Value channels in (8, 12, 3) to have a better result in the searching step.

Hence, we present in Table 4 the comparison results for an aerial localisation task. In the table, we expose the information on the poses inside the entire path, the number of testing images, and the accuracy of each method. This accuracy consists of the number of correctly re-localised images from acquiring the sub-map to the corresponding pose. Our methodology keeps the best localisation results, successfully obtaining more images re-localised on the first two trajectories with an average accuracy of 0.74. In comparison, ORB-SLAM2 performs better on the last two trajectories with an average accuracy of 0.81.

This evaluation indicates that our methodology wins in the first two trajectories while the SLAM system obtains more

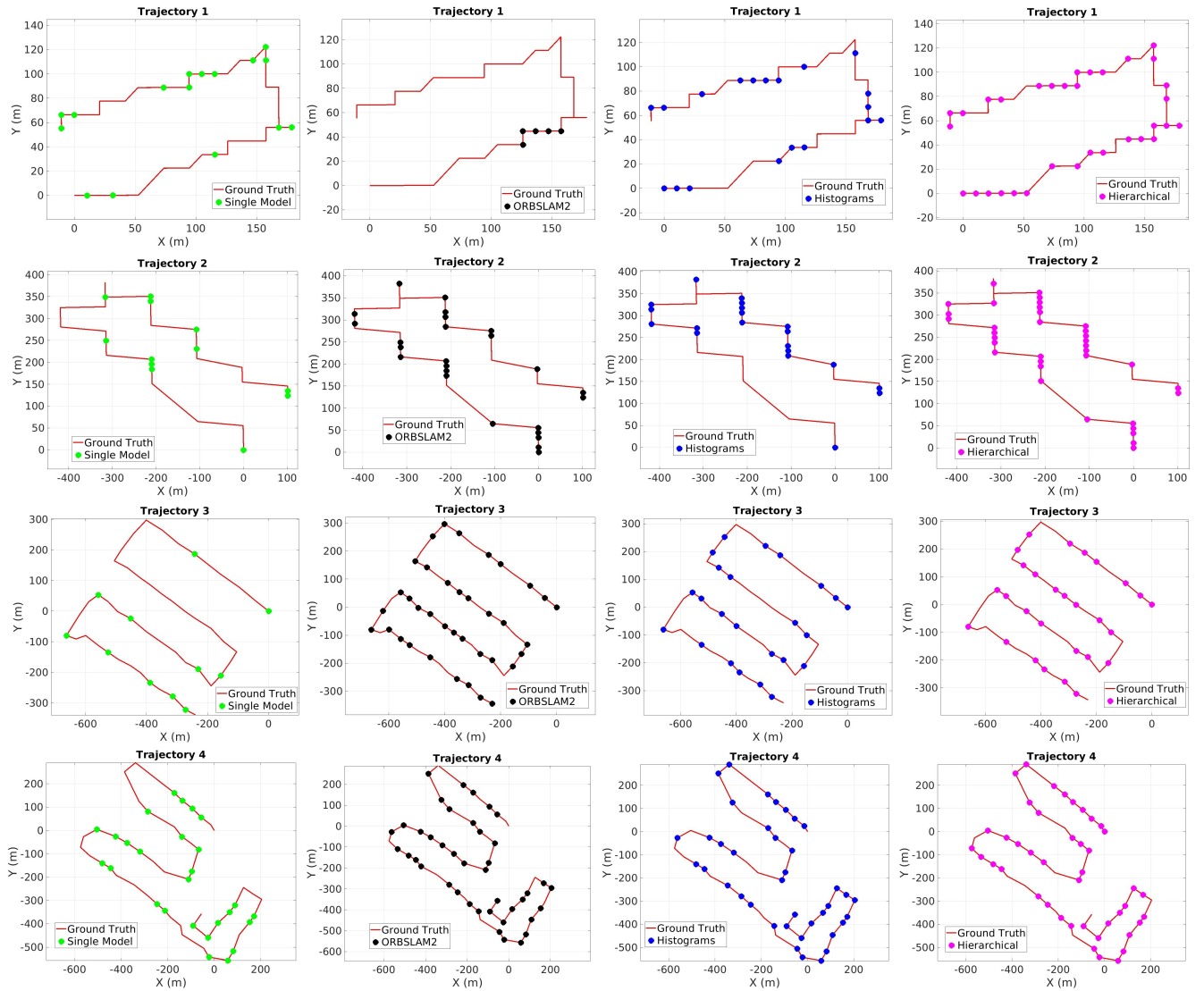


Figure 7: We utilised test images to re-localise and establish the ground truth based on the training trajectory. The circles in the results represent the recovered poses for each evaluated method. The first column corresponds to the evaluation using a single model, the second with ORB-SLAM2, the third using a histogram colour method, and the last with our hierarchical approach.

poses in the last two. Nevertheless, we present a similar result to SLAM, arguing that localisation is achieved in these trajectories. For the latter, the SLAM system drops its performance in complicated scenarios lacking descriptors or with a repetitive pattern, such as trajectories 1 and 2. In contrast, a deep learning methodology keeps the accuracy enough to establish the localisation, learning the essential features. Nonetheless, the results with a single model demonstrate our assumptions regarding catastrophic forgetting of the first data. Even with a continual learning strategy, the network loses knowledge as more information arrives.

The results of the learning method with a single model demonstrate a maximum accuracy of 0.43. Conversely, the methodology using a keyframe search system with a colour

histogram improves pose acquisition to below 0.67. Nevertheless, the similarity of several images in colour affects this outcome, particularly in the first two trajectories. Thus, a colour histogram and feature extraction may not be advantageous, particularly in complex trajectories. Therefore, a multi-model-based system has the potential to alleviate the burden of rigid and data-sensitive training by enabling the learning of poses to be split into different models.

As a final result, we present the camera poses recovered with each comparison method using a testing trajectory similar to the training. Thus, if an aerial image is re-localised, we obtain the sub-map and its pose close to the ground truth. Figure 7 shows the trajectories results and poses returned using each method. We can see that in some cases, there are empty

Table 4: Accuracy results for re-localisation poses using comparison methods.

Trajectory	Poses	Images	Single Model	Histograms	ORB-SLAM2	Hierarchical
1	47	144	0.2500	0.3055	0.1041	0.7083
2	50	117	0.2564	0.3076	0.5897	0.7777
3	50	84	0.2976	0.6071	0.9166	0.8928
4	52	607	0.4382	0.6690	0.7166	0.7001

Table 5: Fps results using the comparison methods.

Approach	Traj. 1	Traj. 2	Traj. 3	Traj. 4
Single model	55.30	48.28	55.64	62.76
Histogram	59.63	62.40	64.80	61.87
ORB-SLAM2	85.47	83.33	92.57	89.28
Hierarchical	77.44	68.52	67.63	63.37

spaces in the trajectories, and this is because the method can't obtain the correct localisation in that zone. In addition, to analyse the response time of each method, we present in Table 5 the processing speed in fps, where ORB-SLAM2 obtains a higher speed but not so far from that obtained with our approach.

Finally, our method outperforms the others when considering the number of correctly localised images, as shown in Table 6. However, ORB-SLAM2 demonstrates superior re-localisation performance for trajectories 3 and 4 but not for the earlier ones. We argue that our methodology could be helpful for inspection tasks and as a backup localisation method in case the GPS signal is lost, as it obtains a greater number of images re-located even in the last trajectories.

Table 6: Comparison of the images used to retrieve the poses by each method in the four flight trajectories.

Approach	Traj. 1	Traj. 2	Traj. 3	Traj. 4
Single model	36	30	25	225
Histogram	44	36	51	266
ORB-SLAM2	15	69	77	435
Hierarchical	102	91	75	425

5 CONCLUSION

We have presented a hierarchical continual learning approach to aerial localisation using two networks and a concept based on sub-mapping. In addition, we have developed a multi-model process for each sub-map divided into the entire trajectory, with information on the flight coordinates. The methodology aims to identify a sub-map that best represents the course using the InceptionV4 network to determine the image location and load the corresponding model to obtain the pose. To achieve this, we have continuously trained the

networks during a flight mission using the latent replay strategy while storing representative keyframes of the sub-map to determine which model to load. As a result, this methodology offers the advantage of maintaining good accuracy while reducing the computational resources in the training step.

In addition, to alleviate catastrophic forgetting, we proposed dividing the training into multiple training models, each with information on five camera poses. Our approach outperforms the localisation results compared to a single model training and a methodology based on the colour histogram. Nonetheless, we maintain enough accuracy with the ORB-SLAM2 system. Moreover, we have demonstrated this approach as a backup localisation for UAVs with an average accuracy of 0.77 and a performance speed of 69 fps, which is sufficient for real-time systems. For future work, we will improve the methodology to inspire new techniques based on autonomous navigation, localisation and vision tasks using continual learning strategies, including a regression network to estimate poses instead of getting the label class.

ACKNOWLEDGEMENTS

The first author is thankful for his scholarship funded by Consejo Nacional de Humanidades, Ciencia y Tecnología (CONAHCYT) under 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] 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.
- [3] 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.
- [4] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015.

- [5] Aldrich A Cabrera-Ponce and J Martinez-Carranza. Aerial geo-localisation for mavs using posenet. In *2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS)*, pages 192–198. IEEE, 2019.
- [6] Ghasem Abdi, Farhad Samadzadegan, and Franz Kurz. Pose estimation of unmanned aerial vehicles based on a vision-aided multi-sensor fusion. In *XXII ISPRS Congress, Technical Commission I*, volume 41, pages 193–199, 2016.
- [7] Ruben Mascaro, Lucas Teixeira, Timo Hinzmann, Roland Siegwart, and Margarita Chli. Gomsf: Graph-optimization based multi-sensor fusion for robust uav pose estimation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1421–1428. IEEE, 2018.
- [8] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual-inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.
- [9] Diego Benjumea, Alfonso Alcántara, Agustin Ramos, Arturo Torres-Gonzalez, Pedro Sánchez-Cuevas, Jesus Capitan, Guillermo Heredia, and Anibal Ollero. Localization system for lightweight unmanned aerial vehicles in inspection tasks. *Sensors*, 21(17):5937, 2021.
- [10] Andrea Vallone, Frederik Warburg, Hans Hansen, Søren Hauberg, and Javier Civera. Danish airs and grounds: A dataset for aerial-to-street-level place recognition and localization. *IEEE Robotics and Automation Letters*, 7(4):9207–9214, 2022.
- [11] Akshay Shetty and Grace Xingxin Gao. Uav pose estimation using cross-view geolocalization with satellite imagery. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1827–1833. IEEE, 2019.
- [12] Aleksis Pirinen, Anton Samuelsson, John Backsund, and Kalle Åström. Aerial view goal localization with reinforcement learning. *arXiv preprint arXiv:2209.03694*, 2022.
- [13] Noe Samano, Mengjie Zhou, and Andrew Calway. Global aerial localisation using image and map embeddings. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5788–5794. IEEE, 2021.
- [14] Shitao Tang, Chengzhou Tang, Rui Huang, Siyu Zhu, and Ping Tan. Learning camera localization via dense scene matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1831–1841, 2021.
- [15] Abhinav Valada, Noha Radwan, and Wolfram Burgard. Deep auxiliary learning for visual localization and odometry. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6939–6946. IEEE, 2018.
- [16] Hunter Blanton, Connor Greenwell, Scott Workman, and Nathan Jacobs. Extending absolute pose regression to multiple scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 38–39, 2020.
- [17] MS Müller, S Urban, and B Jutzi. Squeeze-pose: Image based pose regression with small convolutional neural networks for real time uas navigation. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4:49, 2017.
- [18] Aldrich A Cabrera-Ponce and Jose Martinez-Carranza. Convolutional neural networks for geo-localisation with a single aerial image. *Journal of Real-Time Image Processing*, 19(3):565–575, 2022.
- [19] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.
- [20] Xiaofan Yu, Yunhui Guo, Sicun Gao, and Tajana Rosing. Scale: Online self-supervised lifelong learning without prior knowledge. *arXiv preprint arXiv:2208.11266*, 2022.
- [21] Kevin Thandiackal, Tiziano Portenier, Andrea Giovannini, Maria Gabrani, and Orcun Goksel. Generative feature-driven image replay for continual learning. *arXiv preprint arXiv:2106.05350*, 2021.
- [22] 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.
- [23] 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.
- [24] Robin Karlsson, Alexander Carballo, Keisuke Fujii, Kento Ohtani, and Kazuya Takeda. Predictive world models from real-world partial observations. *arXiv preprint arXiv:2301.04783*, 2023.
- [25] Xingguang Zhong, Yue Pan, Jens Behley, and Cyrill Stachniss. Shine-mapping: Large-scale 3d mapping using sparse hierarchical implicit neural representations. *arXiv preprint arXiv:2210.02299*, 2022.

- [26] 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.
- [27] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. imap: Implicit mapping and positioning in real-time. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6229–6238, 2021.
- [28] 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.
- [29] Ali Safa, Tim Verbelen, Ilja Ocket, André Bourdoux, Hichem Sahli, Francky Catthoor, and Georges Gielen. Learning to slam on the fly in unknown environments: A continual learning approach for drones in visually ambiguous scenes. *arXiv preprint arXiv:2208.12997*, 2022.
- [30] 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.

Determining the moment of inertia tensor of a UAV using a motion capture system

D. Yang, S. Al-Zubaidi, K. Stol*
University of Auckland, 20 Symonds Street, Auckland

ABSTRACT

This article proposes a novel method for finding the mass moment of inertia of a small Unmanned Aerial Vehicle (UAV). The method relies on a motion capture system to record the orientation of the object-under-test oscillating from a compound pendulum. The novelty is that it determines the location of the centre of mass and the full inertia tensor from the same set of experiments. The motion capture system records the object-under-test orientation when it is stationary and oscillating from the pendulum. By doing this for multiple orientations, the process can accurately determine the location of the centre of mass to $\pm 2\text{mm}$ error and the full moment of inertia tensor of the object to an average 3.4% error for objects with inertia in the order of 10^{-2}kgm^2 .

1 INTRODUCTION

Mass moment of inertia (MoI) is crucial in many dynamical systems, such as robotic manipulators, aerospace vehicles, and automotive systems. For UAV systems, MoI significantly impacts their flight dynamics and stability. It affects how the UAV responds to external disturbances and changes in attitude. Accurate estimation of UAV attitude is crucial for optimizing UAV performance, ensuring precise control, and achieving desired flight characteristics.

Many methods have been developed to derive the MoI of rigid bodies, including the tabulating method, Computer-Aided Design (CAD) based methods, and pendulum methods. The tabulating method is one of the most straightforward methods that estimate MoI. By tabulating each constituent's inertia as a simplified geometric shape and summing them together with respect to the principle axes while assuming symmetry, a quick and simple model is developed [1, 2]. However, this oversimplification often neglects many complex-shaped components and does not account for products of inertia.

The CAD-based method improves upon the tabulating method by allowing computers to estimate a more comprehensive list of complex components' MoI [3]. However, any deviation in manufacturing of the UAV from the CAD model,

such as imperfect placement of components, results in inaccuracies.

Empirical methods account for these imperfections by directly calculating the MoI of the final product. The most prominent are the torsional and compound pendulum methods.

The torsional pendulum method involves suspending the object of interest on a platform using two or more cables and then twisting it to produce a torsional oscillation, commonly known as bifilar, trifilar, etc., pendulum. The period of oscillation is recorded to determine the MoI of the object about the rotating axis. Although accurate measurements have been achieved with this method [4, 5, 6, 7], its setup can be time-consuming as the placement of the complex object axis needs to align with the platform's throughout the oscillation period. Additionally, complex objects may require additional "ad hoc" support for each measurement when particular axes are to be measured [7]. This additional material further complicates the experiments.

On the other hand, the compound pendulum method suspends the object from a pivot point and measures its unforced swinging period. NASA has employed this method to accurately determine the MoI of various aircraft to $< 2.5\%$ error as early as 1934 [8]. However, this often requires multiple accurate and expensive sensors placed at various locations on the object [9]. More recently, other combinations of measurement instruments have been employed, such as using an inclinometer for obtaining the period of oscillation [10]. However, the authors comment that the instrument is only suitable for measuring slow oscillations; the fast changes result in underestimating the angle, requiring an additional marker-based motion-capturing system to achieve accurate measurements.

These methods rely on the simple principle where a component of gravitational force supplies a constant torque to the system, giving rise to the oscillatory motion. Under the assumption of small angles, it becomes a second-order undamped system. By accurately measuring the period of oscillation, the MoI for that pivot axis can be derived.

However, these methods have limitations, including the reliance on the small angle assumption and undamped oscillation. It should be noted that the dynamics of a simple pendulum motion is a well-explored problem [11, 12, 13]; the damping effect due to both the aerodynamic and structure can be accurately modelled even for large angles.

Furthermore, the major limitation of these methods is the requirement to know the centre of mass (CoM) of the object-

*Email address(es): kstol@auckland.ac.nz

under-test. For instance, the trifilar pendulum necessitates the accurate placement of the object's CoM on the vertical line passing through the rotational centre of the platform. Any misplacement can lead to undesired translations of the entire system, thereby compromising the accuracy of the results. Estimating the CoM can be challenging, particularly for complex-shaped or multi-component objects.

This article proposes a novel method that addresses the abovementioned challenges and provides an accurate and practical solution for measuring MoI in complex-shaped or multi-component objects without requiring precise CoM estimation. This approach takes advantage of the sub-millimetre accuracies of a motion capture system which many laboratories around the world share.

The rest of the paper is organised as follows. Section 2 discusses the high-level approach for this method, followed by a closer examination of system modelling in section 3. Then section 4 describes the experimental setup. The experimental data are analysed and the results discussed in section 5. Finally, the paper concludes and discusses future work in section 6.

2 METHODOLOGY

The fundamental idea follows that of a traditional compound pendulum. The object-under-test, e.g. a MAV, hanging from the pendulum is displaced from its stationary position. Its angular motion is then recorded to derive its MoI.

However, the novelty lies in that the object can be hung from any point on its rigid body in any orientation, and the CoM location is determined in the same experiments as MoI simultaneously using a motion capture system (MoCap). The ideas behind determining both properties are as follows.

When an object is hung from a pendulum with a lightweight pivot arm, its CoM naturally lies below the pivot axis on the vertical plane. By placing the object in various orientations, multiple planes on which the CoM lies can be obtained. Then its position in 3D can be calculated by finding the intersection of all the planes.

A full inertia tensor can be obtained from the oscillation data of various unique orientations. These orientations do not have to be on the principle axes of the object (I_{xx} , I_{yy} , I_{zz}); they can be arbitrary. Each orientation gives us an MoI value about a specific axis. With at least six unique axis values, the full tensor can be derived using linear regression.

The Experimental procedure is explained below:

1. Rigidly attach the object to a pivot arm in an arbitrary orientation. This is the attachment point and object orientation is denoted as one configuration.
2. Start recording this configuration when it is stationary. Allow some time to record this stationary configuration, e.g. 5s, as this stationary data will be used for deriving the location of CoM.

3. When satisfied, give a small displacement to the object such that it oscillates freely for at least 10 periods. This is to allow the MoCap system to capture enough data for processing later. This part of the data (dynamic data) will be used for MoI calculation.
4. After the recording, create a new configuration by hanging it at a different orientation. Note that it is crucial that this new orientation is unique, i.e. not linearly dependent on the previous ones, otherwise, it results in a duplicate measurement of MoI about an already-measured axis.
5. Repeat the steps for at least six recordings for obtaining a full tensor. Additional testing reduces the error of final results.

To process the data, a flow chart describing the high-level process is shown in figure 1. Three groups of data from MoCap are obtained. D_{static} denotes the static data where the system is stationary when hanging freely on a pendulum. D_{rig} and $D_{dynamic}$ are both the dynamic oscillatory data after the system is displaced from its stable equilibrium position; D_{rig} consists of only the pivot arm in oscillation.

The CoM of the object can be obtained with at least $N = 3$ of D_{static} data, where N is the number of experiments conducted. Once the CoM is determined, its distance to the pivot axis can be calculated for each experiment. These distances, in addition to the MoI of the pivot arm obtained from D_{rig} and the MoI of the system from $D_{dynamic}$, can isolate the MoI about an axis through its CoM of the object alone. Then these MoI values are transformed to the full inertia tensor about a set of axes the users defines fixed to the object. Note that at least $N = 6$ is needed since the full inertia tensor has 6 unique values. With this, both the CoM and MoI tensor are derived.

3 THEORY

This section discusses the notations and equations used for calculating both the CoM and MoI of the object.

The object-under-test is denoted as O , the pivot arm that rigidly connects the pivot axis and the object-under-test as A , and the combined system in oscillation S , such that the mass of the combined system m_S is

$$m_S = m_O + m_A \quad (1)$$

where m_O and m_A are the mass of the object and pivot arm, respectively.

3.1 Coordinate frames

Three coordinate frames are defined: world frame $\{W\}$, object frame $\{O\}$, and the pivot arm frame $\{A\}$. Figure 2 visualises the relationship between the frames.

$\{O\}$ is fixed to the body of the object-under-test; its orientation in $\{W\}$ varies from experiment to experiment depending on how the object is attached. The CoM of the object in

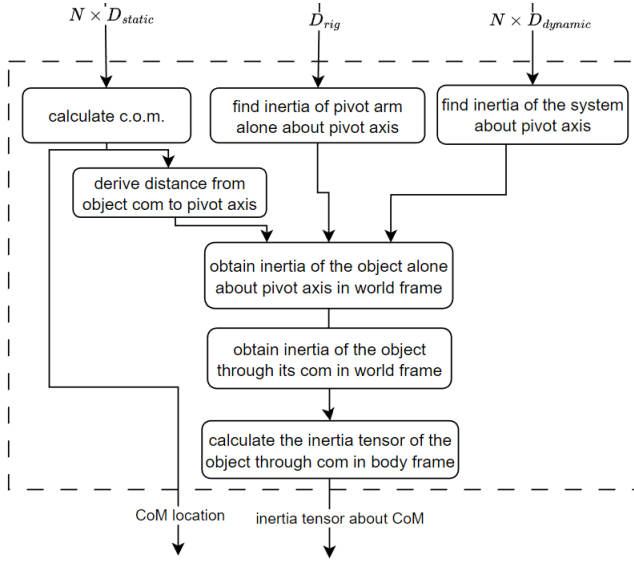


Figure 1: Data processing procedural flowchart, showing the high-level general steps taken towards calculating a rigid body's CoM and MoI

the object frame ${}^O\mathbf{c}$ is unknown but constant throughout all experiments. The pivot arm frame $\{A\}$ is fixed to the pivot arm. Its origin lies at the bearing centre such that its y axis is the pivot axis, which both the object-under-test and the pivot arm oscillate about. $\{A\}$ rotates with the pivot arm. Figure 2 shows a case where the pivot arm's CoM is not directly between the object CoM and the pivot axis. Note that the system CoM always lies directly on the vertical plane below the pivot axis when at rest.

The position of CoM is denoted as $\mathbf{c}$, and other positions are denoted as $\mathbf{p}$. The distance to CoM of a specific item, such as the overall system, S , is noted as $\mathbf{c}_S$; the coordinate frame such measurement is in is noted as ${}^W\mathbf{c}$ in the case of the world frame. For example, ${}^O\mathbf{p}_A$ denotes the position of the arm frame origin in the object frame; ${}^A\mathbf{c}$ denotes the position of the pivot arm's CoM in the arm frame.

The transformation matrices $\mathbf{T}$ of world-to-object ${}^O_W\mathbf{T}$ and world-to-rig ${}^R_W\mathbf{T}$ can be defined as

$${}^i_W\mathbf{T} = \begin{bmatrix} {}^i_W\mathbf{R}_{3 \times 3} & -{}^W\mathbf{p}_i_{3 \times 1} \\ 0 & 1 \end{bmatrix}_{4 \times 4} \quad (2)$$

where $i \in \{O, R\}$, e.g. ${}^O_W\mathbf{R}$ is the rotation matrix from world to object frame.

3.2 Centre of Mass

To find the object CoM location ${}^O\mathbf{c}$, two points on the same plane satisfy the relationship

$${}^O\hat{\mathbf{n}}^T {}^O\mathbf{c}_S = {}^O\hat{\mathbf{n}}^T {}^O\mathbf{p}_A \quad (3)$$

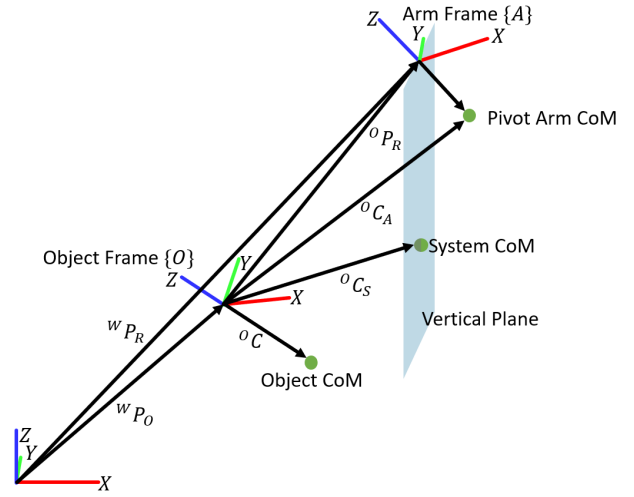


Figure 2: Coordinate Frames. Object frame $\{O\}$ is fixed to the body of the object-under-test in an arbitrary orientation. Arm frame $\{A\}$ rotates with the pivot arm; its origin sits on the pivot axis; its y axis aligns with the pivot axis at all times.

where ${}^O\hat{\mathbf{n}}$ is the normal vector of the vertical plane in the object frame and

$${}^O\hat{\mathbf{n}} = {}^O_W\mathbf{R} {}^W\hat{\mathbf{n}}. \quad (4)$$

The vertical plane can be set to align with a world principle axis, e.g. ${}^W\hat{\mathbf{n}} = [0 \ 1 \ 0]^T$.

As shown in figure 2, ${}^O\mathbf{p}_A$ can be derived from

$${}^O\mathbf{p}_A = {}^O_W\mathbf{R} ({}^W\mathbf{p}_R - {}^W\mathbf{p}_O) \quad (5)$$

where ${}^W\mathbf{p}_R$ and ${}^W\mathbf{p}_O$ are the position of the rig and object frame origins in the world, measured by MoCap. To obtain ${}^O\mathbf{c}_S$, the CoM equation

$$m_t d_t = \sum_i m_i d_i \quad (6)$$

where m_t and m_i denote the total mass of the system and the mass of its constituents, is used. d_t , d_i denote the distance from the system's and its constituents' CoM to a fixed point. Since the system consist of two components: the pivot arm and the object-under-test,

$$m_S {}^O\mathbf{c}_S = m_O {}^O\mathbf{c} + m_A {}^O\mathbf{c}_A. \quad (7)$$

Recall that m_A , m_A and m_O are the mass of the system (Eq.1), pivot arm, and object, respectively. ${}^O\mathbf{c}$ is the CoM position of the object to be solved for. ${}^O\mathbf{c}_A$ can be expressed as

$${}^O\mathbf{c}_A = {}^O_W\mathbf{R} ({}^R_W\mathbf{R}^{-1} {}^A\mathbf{c} + {}^W\mathbf{p}_R - {}^W\mathbf{p}_O) \quad (8)$$

where ${}^A\mathbf{c} = [0, 0, -d_A]^T$, d_A is the distance from the rig's CoM to the pivot axis. d_A is measured by balancing the pivot arm on a sharp edge.

Substituting Eq.7 into Eq.3 gets

$${}^O\hat{\mathbf{n}}^T m_S^{-1}(m_O {}^O\mathbf{c} + m_A {}^O\mathbf{c}_A) = {}^O\hat{\mathbf{n}}^T {}^O\mathbf{p}_A. \quad (9)$$

Rearranging for ${}^O\mathbf{c}$ yields

$${}^O\hat{\mathbf{n}}^T {}^O\mathbf{c} = {}^O\hat{\mathbf{n}}^T ({}^O\mathbf{p}_A - \frac{m_A}{m_S} {}^O\mathbf{c}_A) \frac{m_S}{m_O} \quad (10)$$

where all variables are directly measurable by the MoCap system: ${}^O\mathbf{c}_A$ is described by Eq.8., and ${}^O\mathbf{p}_A$ Eq.5.

With N experiments conducted, a system of equations is obtained as

$$\begin{bmatrix} {}^O\hat{\mathbf{n}}_1^T \\ \vdots \\ {}^O\hat{\mathbf{n}}_N^T \end{bmatrix} {}^O\mathbf{c}_{3 \times 1} = \begin{bmatrix} {}^O\hat{\mathbf{n}}_1 ({}^O\mathbf{p}_{A,1} - \frac{m_A}{m_S} {}^O\mathbf{c}_{A,1}) \frac{m_S}{m_O} \\ \vdots \\ {}^O\hat{\mathbf{n}}_N ({}^O\mathbf{p}_{A,N} - \frac{m_A}{m_S} {}^O\mathbf{c}_{A,N}) \frac{m_S}{m_O} \end{bmatrix}_{N \times 1} \quad (11)$$

where i denotes each of the experimental data. This can be written into the standard form for a system of linear equations $\mathbf{Ax} = \mathbf{b}$, and can solve for $\mathbf{x}$ by finding the pseudoinverse $\mathbf{A}^\dagger$, yielding

$${}^O\mathbf{c} = \begin{bmatrix} {}^O\hat{\mathbf{n}}_1^T \\ \vdots \\ {}^O\hat{\mathbf{n}}_N^T \end{bmatrix}^\dagger \begin{bmatrix} {}^O\hat{\mathbf{n}}_1 ({}^O\mathbf{p}_{A,1} - \frac{m_A}{m_S} {}^O\mathbf{c}_{A,1}) \frac{m_S}{m_O} \\ \vdots \\ {}^O\hat{\mathbf{n}}_N ({}^O\mathbf{p}_{A,N} - \frac{m_A}{m_S} {}^O\mathbf{c}_{A,N}) \frac{m_S}{m_O} \end{bmatrix}. \quad (12)$$

Thus, the least-square error value of ${}^O\mathbf{c}$ is found given the experimental data.

3.3 Moment of Inertia Tensor

To calculate the MoI of an object on a compound pendulum, the equation of motion is

$${}^W I_S \ddot{\theta} + \tau_d + m_s g d_s \sin \theta = 0 \quad (13)$$

where ${}^W I_S$ is the MoI of the system about the pivot axis, θ is the angle of oscillation, $m_s g$ is the gravitational force, and d_s is the distance from the system's CoM to the pivot axis. The τ_d term denotes the damping torque experienced by the system due to both aerodynamics and bearing friction [11]

$$\tau_d = a\dot{\theta} + b\dot{\theta}^2 \operatorname{sgn}(\dot{\theta}) \quad (14)$$

where a, b are the damping constants. Combining Eq.13 and 14 then rearranging to have

$$\ddot{\theta}_k {}^W I_S + a\dot{\theta}_k + b\dot{\theta}_k^2 \operatorname{sgn}(\dot{\theta}_k) = m_s g d_s \sin \theta_k \quad (15)$$

where θ_k is the angle measurement from the MoCap system at time step k . From the dynamic data, a times series of $\theta_k, \dot{\theta}_k, \ddot{\theta}_k$ that can be written into matrix form can be obtained:

$$\begin{bmatrix} \ddot{\theta}_1 & \dot{\theta}_1 & \dot{\theta}_1^2 \operatorname{sgn} \dot{\theta}_1 \\ \vdots & \vdots & \vdots \end{bmatrix}_{K \times 3} \begin{bmatrix} {}^W I_S \\ a \\ b \end{bmatrix} = m_s g d_s \sin \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_K \end{bmatrix}_{K \times 1} \quad (16)$$

where i denotes the value at each corresponding time step and K = the total number of time steps. Solving for $[{}^W I_S \ a \ b]^T$ results in

$$\begin{bmatrix} {}^W I_S \\ a \\ b \end{bmatrix} = m_s g d_s \begin{bmatrix} \ddot{\theta}_1 & \dot{\theta}_1 & \dot{\theta}_1^2 \operatorname{sgn} \dot{\theta}_1 \\ \vdots & \vdots & \vdots \end{bmatrix}^\dagger \sin \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_K \end{bmatrix}_{K \times 1}. \quad (17)$$

The same equations apply to finding the MoI of the pivot arm about the pivot axis ${}^W I_A$. Once ${}^W I_S$ and ${}^W I_A$ are found, the MoI of the object through its CoM can be derived:

$${}^W I = {}^W I_S - {}^W I_A - m_o \| {}^W \mathbf{p}_R - {}^W \mathbf{p}_O \|^2 \quad (18)$$

where ${}^W I$ is the MoI of the object about an axis through its CoM in the world frame. That axis is parallel to and on the same plane as the pivot axis. $m_o \| {}^W \mathbf{p}_R - {}^W \mathbf{p}_O \|^2$ is the parallel axis term. With N experiments, there are N ${}^W I$ values.

To transform these MoI values about the arbitrary axes onto the user-defined object body frame, it can be shown that

$${}^O \hat{\mathbf{z}}^T {}^O \mathbf{I} {}^O \hat{\mathbf{z}} = {}^W I \quad (19)$$

where ${}^O \hat{\mathbf{z}}_{3 \times 1} = [x, y, z]^T$ is the directional vector describing the orientation of the pivot axis in the object frame, and ${}^O \mathbf{I}_{3 \times 3}$ is the full inertia tensor of the object in the object frame.

For N experiments, Eq.19 can be written as

$$\begin{bmatrix} x_i & y_i & z_i \\ \vdots & \vdots & \vdots \end{bmatrix}_{N \times 3} \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} \begin{bmatrix} x_i & \dots \\ y_i & \dots \\ z_i & \dots \end{bmatrix}_{3 \times N} = \begin{bmatrix} {}^W I_i \\ \vdots \\ \vdots \end{bmatrix}_{N \times 1} \quad (20)$$

and expanding the LHS using a standard 2nd-order trinomial expansion yields

$$\begin{bmatrix} x_i^2 & y_i^2 & z_i^2 & 2x_i y_i & 2x_i z_i & 2y_i z_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{N \times 6} \begin{bmatrix} I_{xx} \\ I_{yy} \\ I_{zz} \\ I_{xy} \\ I_{xz} \\ I_{yz} \end{bmatrix} = \begin{bmatrix} {}^W I_i \\ \vdots \\ \vdots \end{bmatrix}_{N \times 1} \quad (21)$$

thus solving for ${}^O \mathbf{I}$ yields

$$\begin{bmatrix} I_{xx} \\ I_{yy} \\ I_{zz} \\ I_{xy} \\ I_{xz} \\ I_{yz} \end{bmatrix} = \begin{bmatrix} x_i^2 & y_i^2 & z_i^2 & 2x_i y_i & 2x_i z_i & 2y_i z_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}^\dagger \begin{bmatrix} {}^W I_i \\ \vdots \\ \vdots \end{bmatrix}. \quad (22)$$

The full tensor of the object-under-test in a frame that is fixed to the object is now derived.

4 EXPERIMENTAL SETUP

This section describes an implementation of the methodology in section 2.

For the purpose of validation, an object of known MoI is used. Shown in figure 3, the object is a Polymethyl methacrylate (PMMA) circular disk of dimension 400mm in diameter, 10mm in thickness and uniform density. The hole cutouts are for the purpose of attachment. The analytical value of the

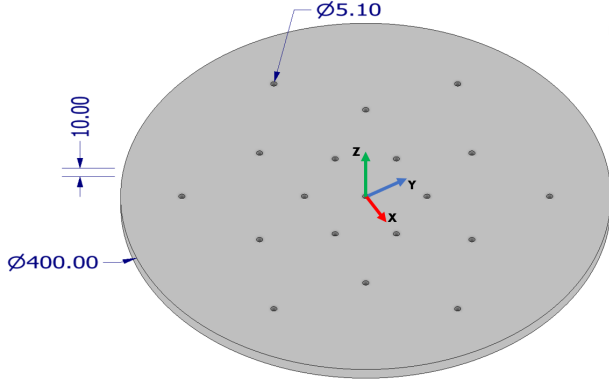


Figure 3: Circular disk PMMA. Dimensions in mm.

full inertia tensor is calculated given the radius, thickness and mass to be

$${}^O\mathbf{I}_{analytical} = \begin{bmatrix} 1.4810 & 0 & 0 \\ 0 & 1.4810 & 0 \\ 0 & 0 & 2.9521 \end{bmatrix} \times 10^{-2} kgm^2 \quad (23)$$

which is in the same order of magnitude as a small UAV [14, 15, 16, 17]. Note that the hole cutouts are also taken into account.

The physical rig setup is shown in figure 4. Two tripods rigidly hold a steel rod horizontally above the ground. They stand a suitable distance apart such that the object-under-test can swing without obstruction. A laser pointer is used to ensure the rod is levelled. Two retro-reflective markers are put on the ends of the rod, marking the pivot axis.

The pivot arm consists of a bearing, two 3D-printed parts, and an aluminium bar. The 3D printed parts are used to rigidly attach the bar to the bearing and the object to the bar. Note that the pivot arm should be constructed in a way such that the distance from its CoM to the pivot axis can be obtained from simple methods such as balancing on a sharp edge accurately. One more marker is placed on the pivot arm such that it tracks the angular displacement of the arm.

For the object-under-test, the object frame is defined by placing 4 markers on the object, marking the origin, +x, +y and +z directions. This origin does not lie on the CoM.

For the motion capture system, VICON cameras with software VICON 3.9.0 is used for recording the object pose. An instance of the recorded objects is shown in figure 5. In the figure, the rig frame is defined such that its x-axis is the pivot axis, and the origin of the frame lies on the leftmost marker. The object frame is fixed to the object-under-test at an arbitrary pose.



Figure 4: An instance of the rig setup

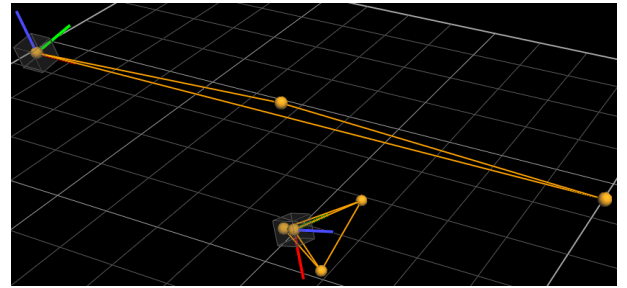


Figure 5: An example of tracked markers.

5 DATA PROCESSING AND RESULT

This section discusses the data processing and visualisation when calculating the CoM and MoI of the object, using MATLAB.

After following the steps described in section 2, VICON exports the data to a list of .csv files and is read by MATLAB. Figure 6 shows one of the experimental data sets. For at least the first 5s, the system is stationary. This section of the data is D_{static} , used for locating the object's CoM. Dynamic data $D_{dynamic}$ starts from 17s in the example in figure 6.

Transformation matrices (Eq.2) are first constructed using D_{static} for each experiment. Figure 7 shows an instance of the object rotation in the world frame (*poseplot()* function). This should reflect the orientation of the object shown in VICON. With all the transformation matrices obtained, the CoM location in the object frame can be derived by finding the point with the least distance to all the intersecting planes with Eq.3.

On the other hand, for $D_{dynamic}$, it is required to obtain the angular velocity and acceleration from the measured angle data by VICON. They are needed for Eq.17. A simple difference function is applied twice to obtain velocity and acceleration data. To resolve the issue of derivative noise amplification, a zero-phase filter function *smoothdata()* in MAT-

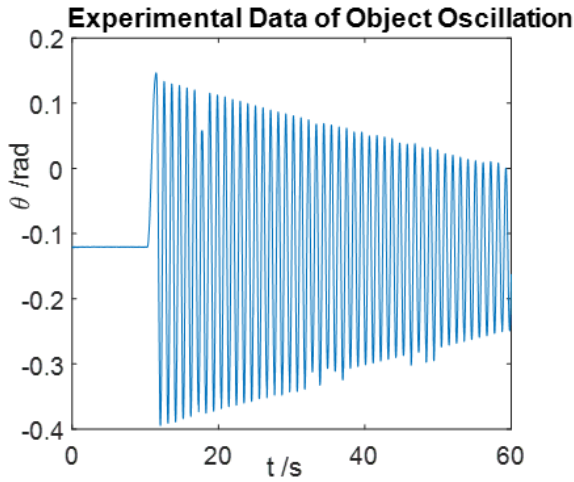


Figure 6: VICON recorded data for one experiment including both static and dynamic data

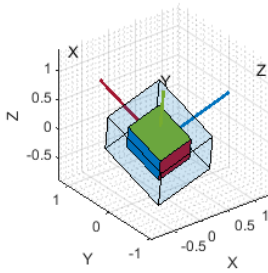


Figure 7: $O_W \mathbf{R}$ visualisation for one experiment. In each experiment, the object frame has a unique orientation in the world frame.

LAB is used with the *method* parameter specified as 'sgolay', denoting a Savitzky-Golay filter. It is good at rejecting particularly high-frequency noise such as those from derivatives [18].

Once the kinematic data are obtained, Eq.8 is applied and finds the system MoI about the pivot axis and the two damping coefficients. The accuracy of the fitted model parameters $[I_S \ a \ b]^T$ can be verified by comparing the experimental data and model-generated data given the initial conditions

$$\begin{bmatrix} \theta_0 \\ \dot{\theta}_0 \end{bmatrix} = \begin{bmatrix} \theta_{peak} \\ 0 \end{bmatrix}$$

where θ_{peak} denotes the angle of oscillation at a peak from the corresponding experimental data. These peaks can be found using the *findpeaks()* function in MATLAB on the filtered data. For the experiments, all the fitted models obtained above 95% match to the experimental data. Figure 9 shows an instance of the compared data.

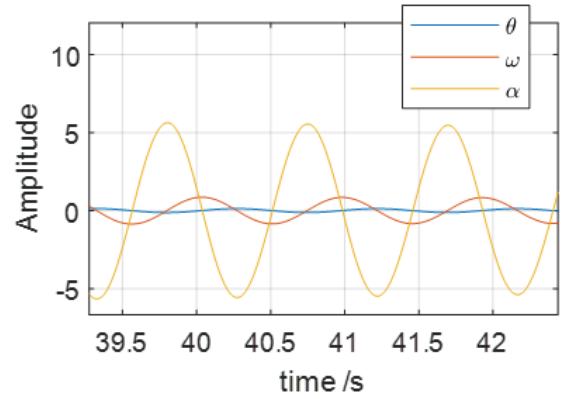


Figure 8: A segment of angle, velocity, and acceleration of the object's oscillation derived and filtered from one experimental VICON data.

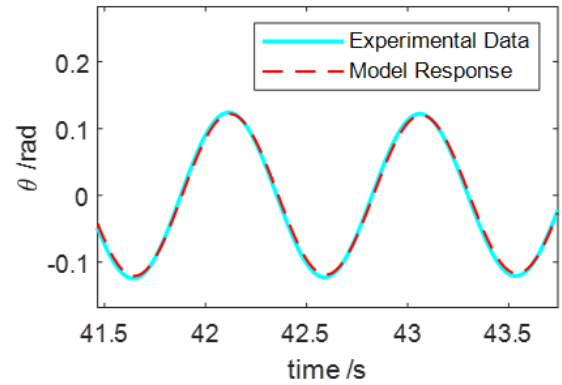


Figure 9: Model validation. By providing the same initial conditions to the fitted model, the model response can be simulated and compared to the experimental data. The figure shows a segment of a 97% match fitted model from one experiment

The CoM value is calculated as, in m,

$$O_{\mathbf{c}} = \begin{bmatrix} -3.75 \times 10^{-4} \\ 4.96 \times 10^{-2} \\ -1.01 \times 10^{-2} \end{bmatrix}$$

Because an object of known geometry is used to validate the experiment, the location of the CoM can be directly measured on the object to be

$$O_{\mathbf{c}_{measured}} = \begin{bmatrix} 0 \\ 5.0 \times 10^{-2} \\ -1.2 \times 10^{-2} \end{bmatrix} m$$

Both the x and y components have less than 1mm error, however, there is a 1.9mm difference in the z axis. This error becomes significant later in the MoI result.

The experimental inertia tensor of the object-under-test in the object frame ${}^O\mathbf{I}$ is found to be, in kgm^2 ,

$${}^O\mathbf{I} = \begin{bmatrix} 0.014871 & 0.00066724 & 0.00029924 \\ 0.00066724 & 0.014910 & -9.8791 \times 10^{-5} \\ 0.00029924 & -9.8791 \times 10^{-5} & 0.031907 \end{bmatrix}$$

Compared with analytical values ${}^O\mathbf{I}_{analytical}$, the percentage error can be calculated

$$\%error = \begin{bmatrix} 0.8\% & - & - \\ - & 1\% & - \\ - & - & 8.2\% \end{bmatrix}$$

Averaging the percentage error on the diagonal terms to obtain an overall error percentage of 3.4%. The experimental values show good estimates in I_{xx} and I_{yy} with a low percentage error of around 1%. However, there's a larger error in I_{zz} of 8.2%; this can be traced back to inaccuracies in the derived CoM position in z , where a large error in the CoM values led to a larger inertia percentage error.

The error of 1.2mm is because there is not enough variation in the third component of the normal plane vector ${}^O\hat{n}$ in Eq.12, i.e. the z axis angle. This made the least-square regression unable to filter out the error in this term. Physically, this means that more different angle placement about the z axis of the object relative to the vertical plane would help with the CoM calculation.

The experiment also shows good estimate about the products of inertia terms. While unable to obtain the percentage error, it can be seen that values are all in the order of magnitude $10^{-4}kgm^2$. Compare to the diagonal terms which are in the order of $10^{-2}kgm^2$, the products of inertia terms are negligible and reflects those in ${}^O\mathbf{I}_{analytical}$. It is worth noting that the calculation does not treat the products of inertia differently from the main diagonal terms, and that the fact that all six value are calculated to be close to that of the analytical values demonstrates the accuracy of the method.

6 CONCLUSION

An experimental method to determine the moment of inertia tensor of a rigid body is proposed in this paper. It also calculates the object's centre of mass location relative to a body frame defined by the markers that the user attaches to the object.

For the object studied, the average percentage error across the three principle axes is calculated to be 3.4%, proving that the method is valid and that the experiments are suitable for obtaining the inertia of objects of this size. To achieve good accuracy, a wide variety of orientations that define new configurations in all three axes is desired.

Future work includes further validating the experiment with asymmetrical objects, as to determine the percentage error of the products of inertia. Additionally, more investigation can go into the sensitivity analysis of the MoI of the object-under-test to the length of the pivot arm. Experiments found

that a long pivot arm increases the parallel axis term in Eq.18 which dominates the system inertia ${}^W\mathbf{I}_S$, making it more difficult to obtain an accurate ${}^W\mathbf{I}$.

ACKNOWLEDGEMENTS

The research reported in this article was conducted as part of "Enabling unmanned aerial vehicles (UAVs) to use tools in complex dynamic environments UOCX2104", which is funded by the New Zealand Ministry of Business, Innovation and Employment.

REFERENCES

- [1] M Yasir Amir and Valiuddin Abbass. Modeling of quadrotor helicopter dynamics. In *2008 International Conference on Smart Manufacturing Application*, pages 100–105. IEEE, 2008.
- [2] Randal W Beard. Quadrotor dynamics and control. *Brigham Young University*, 19(3):46–56, 2008.
- [3] W Hussein, M El-khatib, A Elruby, and H Haleem. Quad rotor design, simulation and implementation. In *International Conference on Computer Science from Algorithms to Applications,(CSAA09)*, 2009.
- [4] Hongwu Wang, Garrett G Grindle, Samuel Connor, and Rory A Cooper. An experimental method for measuring the moment of inertia of an electric power wheelchair. In *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4798–4801. IEEE, 2007.
- [5] Y Minami Koyama, HS Miranda, A Monrroy Cano, and B Senzio-Savino Barzellato. Experimental determination of an irregular object's moment of inertia. In *First International Congress on Instrumentation and Applied Sciences, Cancun QR Mexico*, 2010.
- [6] PL Ringegni, MD Actis, and AJ Patanella. An experimental technique for determining mass inertial properties of irregular shape bodies and mechanical assemblies. *Measurement*, 29(1):63–75, 2001.
- [7] Emiliano Mucchi, Stefano Fiorati, Raffaele Di Gregorio, and Giorgio Dalpiaz. Determining the rigid-body inertia properties of cumbersome systems: Comparison of techniques in time and frequency domain. *Experimental techniques*, 35(3):36, 2011.
- [8] Hartley A Soule and Marvel P Miller. The experimental determination of the moments of inertia of airplanes. Technical report, 1934.
- [9] NASA Armstrong Flight Loads laboratory. X-56 moment of inertia tests. <https://www.youtube.com/watch?v=7xQJ2sVQrUA>.

- [10] Roman Gabl, Thomas Davey, Edd Nixon, and David M Ingram. Accuracy analysis of the measurement of centre of gravity and moment of inertia with a swing. *Applied Sciences*, 11(12):5345, 2021.
- [11] Peter F Hinrichsen. The period of the damped non-linear pendulum. *European Journal of Physics*, 41(5):055002, 2020.
- [12] Kim Johannessen. An analytical solution to the equation of motion for the damped nonlinear pendulum. *European Journal of Physics*, 35(3):035014, 2014.
- [13] Augusto Beléndez, Carolina Pascual, DI Méndez, Tarcisio Beléndez, and Cristian Neipp. Exact solution for the nonlinear pendulum. *Revista brasileira de ensino de física*, 29:645–648, 2007.
- [14] Jemie Muliadi, Rizki Langit, and Benyamin Kusumoputro. Estimating the uav moments of inertia directly from its flight data. In *2017 15th International Conference on Quality in Research (QIR): International Symposium on Electrical and Computer Engineering*, pages 190–196. IEEE, 2017.
- [15] Elisa Capello, Hyeongjun Park, Bruno Tavora, Giorgio Guglieri, and Marcello Romano. Modeling and experimental parameter identification of a multicopter via a compound pendulum test rig. In *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, pages 308–317. IEEE, 2015.
- [16] Valentin Wüest, Vijay Kumar, and Giuseppe Loianno. Online estimation of geometric and inertia parameters for multirotor aerial vehicles. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1884–1890. IEEE, 2019.
- [17] Michael Burri, Janosch Nikolic, Helen Oleynikova, Markus W Achtelik, and Roland Siegwart. Maximum likelihood parameter identification for mavs. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4297–4303. IEEE, 2016.
- [18] Çağatay Candan and Hakan Inan. A unified framework for derivation and implementation of savitzky–golay filters. *Signal Processing*, 104:203–211, 2014.

Why do I need to speak to my drone?

Jose Martinez Carranza*, Delia Irazu Hernandez-Farias, Leticia Oyuki Rojas-Perez, and Aldrich Alfredo Cabrera Ponce†

Instituto Nacional de Astrofisica Optica y Electronica (INAOE), Puebla, Mexico

† Benemerita Universidad Autonoma de Puebla (BUAP), Puebla, Mexico

ABSTRACT

Seeking to enhance the autonomous performance of a delivery drone, which is envisioned for delivering parcels to homes, offices, and other urban and rural locations, we present a study where we evaluate the use of state-of-the-art Artificial Intelligence (AI) tools, specifically language and generative models such as Diffusion Models, CLIP, and ChatGPT, to address the challenge of recognising unfamiliar places based on textual descriptions provided by customers. This concept draws inspiration from contemporary e-commerce practices, where customers typically provide textual descriptions of delivery destinations. Such descriptions serve as additional guidance to assist the delivery person in reaching the intended location. Our objective is to investigate how to process such textual descriptions to be comparable with the observations made by a delivery drone. Based on our preliminary assessment, these AI models can leverage the autonomous behaviour of a drone that is required to navigate to unfamiliar destinations, relying on images captured by its onboard camera and the textual description provided by the customer to accomplish this goal.

1 INTRODUCTION

When the Amazon Primer Air program was launched in December 2013, the expectations for drone delivery rapidly grew. Today, the technology is ready, and drones can arrive at a location autonomously using a GPS coordinate to guide them toward the destination. Unfortunately, in USA, as much as in most of the countries of the Commonwealth, the regulations have inhibited the widespread use of this technology [1], and that has made it difficult to carry out a realistic evaluation of how effective this technology would be in real life delivery scenarios, especially in developing countries where urban organisation tends to be less ordered, let alone rural areas.

Poorly designed and maintained neighbourhoods could complicate the transportation of goods, especially in the stage

known as the last-mile delivery, where parcels are taken from a distribution centre to the last destination provided by the customer (home or business address) [2]. The last-mile delivery is expected to be the most challenging and costly stage of the logistics behind the delivery of goods in e-commerce [3].

Among the different problems in last-mile delivery, an inaccurate or wrong delivery address is a recurrent one, especially when the customer is away from home to assist the courier [4]. One can expect this problem to worsen if the courier is unfamiliar with the area or has never been to the delivery place. For these reasons, it is a common practice to request the customer with a textual description of the delivery place, soliciting references or landmarks that could help the courier recognise the delivery location more quickly. Under this scenario, even a description of the appearance of the place and objects around it, such as trees, cars, and lampposts, are expected to be provided by the customer if they consider that these are distinctive landmarks that would help to recognise their place.

If, in the near future, companies plan to utilise autonomous drones to assist in the last-mile delivery stage, then we can anticipate that these delivery drones will encounter the same aforementioned issues, in particular, the recognition of the places where it has never been before. In this respect, a delivery drone could also exploit the customer's textual description the same way a delivery person uses it to find a delivery place. However, to achieve this, the textual information has to be represented so that it can be compared with observations of the scene made by the delivery drone. As it has been widely investigated, onboard cameras are low-cost sensors that can produce rich visual information [5], and it has become common now for a drone to have at least one onboard camera.

Therefore, a broader question is, what is the shared space where textual and visual information can be mapped such that these can be compared in order to recognise that what is being described is similar to what is being visually observed? Motivated by the latter, in this work, we explore the use of state-of-the-art language and generative models to address this question. First, we assume that a customer provides a textual description of a delivery place (denoted as *target*). Then, we propose to use a generative model known as the Stable Diffusion model [6], to generate images from a textual description that can be compared directly against the target. Figure 1 shows some examples of images generated by the Diffusion Model from textual descriptions provided by a subject. However, like any other generative model, an image gener-

*Email address: carranza@inaoe.mx



Figure 1: Target images and their synthetic images generated with Stable Diffusion using prompts provided by human subjects in this study. Each column shows the best synthetic image generated with a prompt provided by subject 1 in this study, in total, 10 prompts. The best image has the highest similarity score that compares the image embeddings of the target and synthetic images using the cosine distance.

ated with Stable Diffusion can not be compared directly with the image of the delivery place at the pixel level such as it is typical for place recognition methods based on visual descriptors or neural networks [7, 8]. For instance, if the customer inputs “a red car”, then the Diffusion Model is likely to generate a red car visually dissimilar to a red car observed in the drone’s camera image.

A visual comparison may not be possible between an artificially generated image and a real camera image. However, at the semantic level, if a red car exists, we want a similarity measure to reward it. Thus, the more semantic concepts are found between these images, the higher the similarity score. Therefore, we propose to explore the use of the Contrastive Language-Image Pretraining model (CLIP) [9], a neural network model that has learned to associate images with textual descriptions through the generation of numerical representation known as *embedding*, where images and textual descriptions are represented in a shared space at the semantic level.

The Diffusion Model used in this work is still slow to be used in real time. Nevertheless, we have designed a set of experiments to get insights into its potential use and discuss some avenues of research on how it could be exploited in tandem with language models such as CLIP and ChatGPT.

2 RELATED WORK

Recently, there is a strong trend in developing multi-modal approaches which aim to take advantage of more than one modality of information (visual, textual, speech, etc.).

Among these, there is one combining the capabilities of Computer Vision and Natural Language Processing commonly denoted as *Vision-and-Language models*¹ which covers many very challenging tasks such as i) *image captioning* aiming to generate a natural language description from an input image, ii) *visual question-answering* which objective is to find answers by means of a question in natural language and a related image, iii) *image retrieval* which aims to retrieve the data in a given modality by the cues provided in another modality, iv) *Phrase grounding* which involves object detection from an input image and a phrase in natural language, and v) *image generation* which aims to generate an image from the information provided by a textual description in natural language.

Generating an image from a textual input is a challenging task that has been addressed from different perspectives. Generative Adversarial Networks (GANs) have been widely exploited for image generation demonstrating an impressive performance [10]. Transformers-based architectures, which have proven to be very powerful for addressing several natural language processing tasks, have also been used as a way to generate images from textual descriptions [11]. State-of-the-art has been outperformed by Diffusion models², which are inspired by non-equilibrium thermodynamics [12]. In particular, some of these models have attracted attention beyond the research community as they have been made publicly available online through simple interfaces causing their extensive

¹https://huggingface.co/blog/vision_language_pretraining

²<https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

use in many respects. Examples of these models are *Imagen*³ and *DALLE-2*⁴. Another model of this type is *Stable Diffusion*⁵ [6] which besides allows the users to access the source code and weights of the model⁶.

There are other visual-and-language models capable of performing more general tasks. CLIP allows us to determine if an image and a textual description can be paired since it was pre-trained in “image-text” pairs. For doing so, both inputs are converted to an embedding representation allowing us to compare them by means of cosine similarity. On the other hand, there is VisualBERT a model for learning joint contextualised representations of vision and language [13] enabling them to capture semantics between images and texts.

A very important aspect to be considered in *text-to-image* tasks is how to evaluate the model’s performance. Some options are to quantitatively measure the alignment between the images generated and the textual descriptions or to measure if the objects mentioned in the textual description are detected in the generated image [14]. However, the inherently subjective nature of textual descriptions made in natural language has made it important to assess these systems by considering human judgments on the generated images [15].

Vision-language models can have a wide range of applications [16] in different AI areas like robotics, where language and vision can provide very useful information for improving the understanding of the environment where a robot must perform. However, related work combining vision, language, and robotics is scarce. Vision-and-Language Navigation attempts to provide communication skills between humans and agents for navigating in 3D environments [17]. The use of language has also been used as a tool for monocular depth estimation by means of CLIP [18] and also by processing spoken language together with object recognition [19]. Taking advantage of DALL-E, in [20] DALL-E-Bot is presented, it is an autonomous robot able to rearrange objects in a scene by inferring a textual description, then generating an image representing a human-like arrangement of these objects, and finally performing physically the movements.

To the best of our knowledge, diffusion models have not been used for helping on autonomous drone navigation. Our proposal is aimed to contribute in this direction by evaluating the performance of such models to generate images from a textual description provided by users and enable a direct comparison with target images that could be captured with the drone’s onboard camera.

3 METHODOLOGY

The CLIP model represents an image or a textual description via a numerical embedding vector that can be compared using a similarity score based on the cosine distance (with

values between -100 and 100). In a straightforward manner, we could compute the embedding of the textual description provided by the customer and compare it against the computed embedding of a target image. We explored this approach in the first part of the experiments, and it will be noted that the similarity score between a textual description and a target image reaches an average of 30. In contrast, two embeddings of the same image achieve a score of 100 and if the image begins to change in appearance, the less similar, the smaller the score with a tendency towards zero.

We also observed that different textual descriptions with significant changes for the same target image produced scores with no significant difference. This would make it difficult to assess whether a textual description is better than another or the opposite, this is, whether one image corresponds better to a textual description than another. For this reason, we propose to map the textual description to the visual space through the use of the Stable Diffusion model. Thus, an image embedding can be computed with CLIP and compared directly with the corresponding image embedding of the target image. In this work, textual and image embeddings are numerical vectors of 512 float numbers.

For the comparison, we employ the cosine distance between two vectors as proposed in the CLIP methodology. Suppose we have an embedding for a target image $\mathbf{e}_t$ and an one for the image generated with Stable Diffusion $\mathbf{e}_s$, then the similarity score is (Note that for the sake of interpretation, the cosine distance is multiplied by 100):

$$score = 100.0 \times \cos(\mathbf{e}_t, \mathbf{e}_s) = 100.0 \times \frac{\mathbf{e}_t \cdot \mathbf{e}_s}{\|\mathbf{e}_t\| \|\mathbf{e}_s\|} \quad (1)$$

In sum, given a textual description, we use Stable Diffusion to generate a synthetic image, which is expected to contain semantic information described in the text and this is why we can compare it against a target image at the semantic level using CLIP.

4 EXPERIMENTAL FRAMEWORK

To evaluate our proposal, we assume that we have target images depicting different outdoor areas representing delivery locations, and then we asked subjects to provide textual information about the target images by only looking at them. The subjects had not seen the images before, and only at the moment of the request can they pay attention to the image and write down their description. We seek to emulate the scenario where a customer can visually observe a delivery place and describe it in text. Then, the AI tools can use this textual description to generate an image that will resemble what the subject is visualising so that the delivery drone can use such an image to search out the delivery place even if it has never been there before.

For our experiments, we pulled out 5 images from the internet with no particular appearance in mind: 1) a house with

³<https://imagen.research.google/>

⁴<https://openai.com/dall-e-2>

⁵<https://stablediffusionweb.com/>

⁶<https://github.com/CompVis/stable-diffusion>

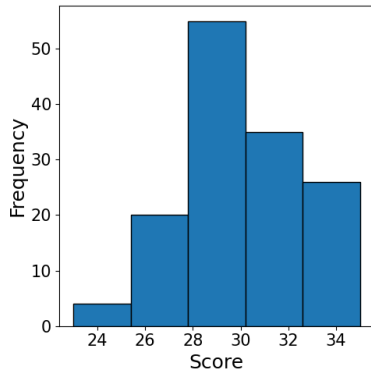


Figure 2: Distribution of scores obtained with text embeddings from subject prompts compared to target image embeddings.

cars parked in front of it; 2) a food truck selling hot dogs; 3) a kiosk; 4) a basketball court; 5) a house with a swimming pool. Next, we asked three subjects to look at these images in order to provide textual descriptions of the places, which we will refer as “prompts”. It is important to mention that the subjects never used *text-to-image* generative models before and that they were not provided with any guidelines for doing this task. Each prompt is passed to the Stable Diffusion model tuned to generate 10 images per prompt to have more than one synthetic image per prompt. Under these settings, we propose two types of prompt generation: i) subject 1 will provide 10 prompts for each one of the target images before using Stable Diffusion; ii) subject 2 and 3 will provide a prompt waiting to see the 10 synthetic images generated by Stable Diffusion and use it as feedback to improve the following prompt; they will continue this iteration until 10 prompts for each target image have been written. This variation in the prompt generation is intended to observe whether Stable Diffusion can improve its output through interaction with subjects 2 and 3; or whether just good prompt writing is enough. Note that for each subject, 10 prompts per target image will be written down, thus generating a total of 100 images per target image. Finally, in a step forward towards what is known as “prompt engineering”, we also propose to use ChatGPT-4, a Large Language Model (LLM) [21], to generate the best textual description possible out of the 10 prompts generated by each subject for each target image and pass it to Stable Diffusion. The goal is to assess whether a powerful LLM can contribute to the generation of better synthetic images.

4.1 Comparison using Text Embedding

Before describing the results obtained with images generated with prompts via Stable Diffusion, and as mentioned in Section 3, we evaluated a direct comparison of the prompts given by the subject against the target images. Our goal was to assess the capacity of the text embedding to represent a

textual description of a place rather than just short sentences describing a few objects, as it has been a common practice in previous works using CLIP. It is worth noticing that the average number of words in subject prompts is 32 words; CLIP fails to compute an embedding for prompts larger than 80 words. Thus, embeddings for the 150 prompts generated by the subjects were computed and compared against their corresponding target image using the similarity score (Eq. 1). The distribution of the scores is shown in Figure 2. Note that the average value is around 30, with 23 for the lowest and 35 for the highest, which makes it difficult to assess whether one textual description could be better than another if these are directly compared against the image embedding. Hence, we propose comparing image embeddings where visual information may encode better semantic information.

4.2 Results from Subject Prompts

Figure 1 presents a mosaic of the best images generated by Stable Diffusion, as determined by the similarity scores in comparison to the target image (displayed in the first column). These images were produced using prompts from Subject 1. The subsequent column showcases the best image from the ten synthetic images generated by the first prompt. In a similar fashion, the next column represents the best image from the second prompt, and this pattern continues. Each row across all prompts represents the best synthetic images corresponding to each target image. Upon observing these images, one can note their striking similarity to the target image. This indicates that Stable Diffusion when paired with CLIP, is capable of generating and identifying viable candidate images, potentially useful in delivery location recognition, as demonstrated by their visual quality.

Due to the lack of space, we cannot include the respective mosaics for subjects 2 and 3, but we can state that the quality in similarity holds. We can summarise the score distribution of the 100 synthetic images generated for each target image per subject as shown in Figure 3. Establishing which subject produced the best prompts is complicated since the distributions resemble each other. Note that subject 1 created very good prompts for the fifth target image, the house with a pool, whereas subject 3 did it for the food truck, the second target image. We must recall that subject 1 wrote their prompts first and then used Stable Diffusion, and subjects 2 and 3 produced a prompt, had a look at the resulting 10 synthetic images, and used their judgment to propose a following prompt, seeking to improve the next batch of synthetic images. However, it seems this did not have a significant effect and is worth investigating with more subjects and images in a future study. Meanwhile, we should highlight that the best values reach around 80 and, for some images, a score of 90. By looking at the distributions, one may think that the synthetic images exhibit a score of 70 on average. Nevertheless, the important fact is that with our proposed methodology, we are able to generate and identify synthetic images with scores

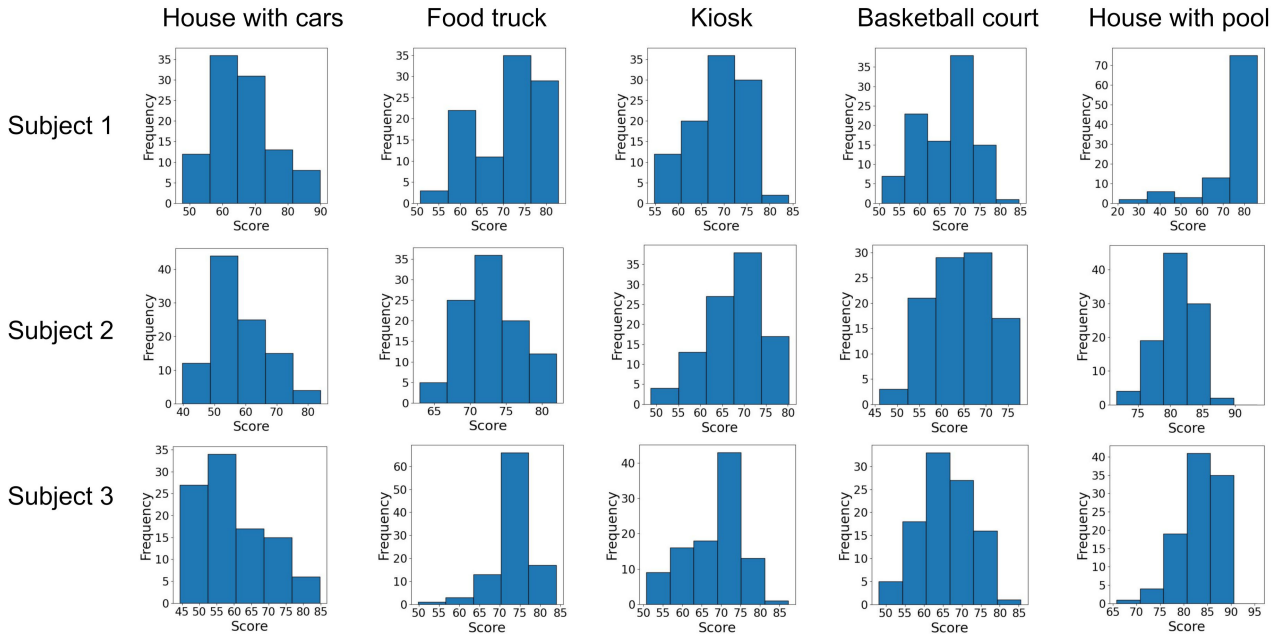


Figure 3: Score distribution for all the images generated with Stable Diffusion using the subject prompts for each target image.

bigger than 80 with a notable visual similarity.

Inspired by the literature assessing the performance of generative models by means of human judgements, we also asked all subjects to choose the image that, in their opinion, was most similar to the target image. To do this, given that for each prompt, Stable Diffusion generates 10 images; then the subject selects 1 out of these 10 images marking it up as the most similar to the target image before applying the evaluation with the CLIP embeddings. We found that the user coincided in 30% with the most similar image found via CLIP embeddings and the similarity score. However, if we take the selected image by the user with the highest score out of the 100 images per target image, then this percentage increases to 47%.

To summarise and illustrate the above, for each target image and using the data of subject 1, Figure 4 shows the image selected by the user with the highest score shown under the “Subject 1 Selected” column and the prompt passed to Stable Diffusion to generate it. The image with the highest score is shown under the “CLIP” column and the corresponding prompt. It stands out that in 3 out of 5 images (60%), subject 1 coincides in their choice of the most similar image with the highest score. Moreover, the similarity between images is also notable for the other two target images. For completeness but also to save space, Figure 5 only shows the images in the same fashion, this is, those selected by the user versus that of the highest score for each target image. Note that in some cases, the resemblance between the synthetic and target images is uncanny.

4.3 Subjects’ Prompt Analysis

The rapid widespread use of generative models (like the ones mentioned in Section 2 or even ChatGPT-4) has led to the emergence of a new research area denoted as “prompt engineering”. It involves the practice and skills for writing effective prompts for generative models. A very important aspect of obtaining the desired images from a textual description is the ability to select the correct words. However, this is not the only key to warrant the performance of generative models, other aspects like a certain format and style, adding keywords, and key phrases have been recognised as a way to improve prompts. People having more experience with writing prompts usually execute the prompt, assess the result, and try to adapt the prompt to improve the obtained output [22]. These modifications are often called “prompt modifiers” a taxonomy of six different types of them is proposed in [23].

Despite its implications for improving the use of generative models, there are only a few available resources helping and guiding users on these tasks for example on the use of OpenAI’s documentation⁷ or the online templates created by the AI-generated art community [24]. Unfortunately, for a task like the one addressed in this paper, there are no available resources for guiding in a prompt generation.

As already mentioned, in the experiments carried out, we rely on the subjects’ backgrounds for creating the prompts describing the target images. A qualitative analysis on the basis of two of the types of “prompt modifiers” in the aforementioned taxonomy⁸ was done. All the prompts contain at least

⁷<https://platform.openai.com/examples>

⁸The remaining aspects were not considered since they are strongly re-

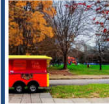
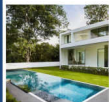
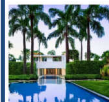
Target Image	Subject 1 Selected			CLIP			ChatGPT-4		
	Prompt	Image from Stable Diffusion	Score	Prompt	Image from Stable Diffusion	Score	Prompt	Image from Stable Diffusion	Score
	"Five white cars parked in front of a luxury house with a bricks entrance and tile roof, sunny days and there are palm trees"		87.3%	"Five white cars seem to go out from a luxury house with a bricks entrance and a tile roof, the shadow of a palm tree is next to one of the cars"		89.8%	"Sunny day with a few clouds over a luxury, yellow house with a brick entrance, tile roof, tall window, and palm trees. Five white cars, including two trucks and a Ferrari, are parked outside."		80.3%
	"In a park with trees without leaves there is a food truck where you can buy hot-dogs, the food truck is red and two people are inside it, no one is waiting for a hot-dog"		82.9%	"In a park with trees without leaves there is a food truck where you can buy hot-dogs, the food truck is red and two people are inside it, no one is waiting for a hot-dog"		82.9%	"In a quiet autumn park, a red hot-dog food truck with yellow letters sits on a cement path flanked by leafless trees and a green grassy area."		80.9%
	"It seems to be the main square of a town where there is a kiosk with arches and a staircase with black handrails at the top of the kiosk there is a dome with arches and glass windows"		84.1%	"It seems to be the main square of a town where there is a kiosk with arches and a staircase with black handrails at the top of the kiosk there is a dome with arches and glass windows"		84.1%	"In a town square, there's a kiosk with yellow arches and pillars. It features a dome with glass windows, a staircase with black handrails, and is near a white tent."		80.7%
	"Nine people are playing basket ball in a court with a green and red floor, the ball is on the floor, at the bottom there are some pine trees"		77.6%	"In the middle of a forest there is a basket ball court where nine people are playing, one guy has red pants and green t-shirt, the people are grouped in two main groups with one group having two people"		79.0%	"In a cloudy day, nine men are playing basketball in a forest park with a green and red court. The orange-yellow ball rests on the floor among the pine trees."		73.5%
	"There is a big grasp and a swimming pool in the front of a white two-floor house with three glass doors, some trees at the bottom"		86.3%	"There is a big grasp and a swimming pool in the front of a white two-floor house with three glass doors, some trees at the bottom"		86.3%	"In a garden with lush trees and palm trees, there's a white, two-floor house with glass doors, an oval-shaped swimming pool, and a grassy area."		86.0%

Figure 4: Images generated with Stable Diffusion using the prompts from subject 1 in this study. We compare the image with most similarity to the target image in three modes: 1) selected by the subject (Selected); 2) with the highest score (CLIP); 3) generated with the prompt produced with ChatGPT-4 whose image score is the highest.

a mention of the most salient "*subject terms*" in the target image (i.e., "pool", "basketball", etc.) and in some cases, these terms were written twice in a given prompt "*repetition*". It is also interesting to note that, one of the subjects who performed an iterative process during prompt generation wrote some consecutive prompts by adding more terms at the end of the phrase, while the prompts generated by the subject who never received feedback from the outcome images are very different between them, in this case, it seems that this subject intentionally tries to describe the target image in a very different way each time.

As mentioned before, we are interested in better taking advantage of prompt engineering for improving the prompts and hence the generated images, we use human-created prompts for feeding the following textual prompts to ChatGPT-4: a) *I have 10 descriptions of a place: [here the list of human-created prompts were added] for each one of the 10 description, could you tell me the average number of words?*; and b) *ok, then given these 10 descriptions, could you mix them to generate the best description whose number of words is around [NUM] words?* (where NUM was replaced by the average length of the prompts generated by each subject). As output, we obtained a new prompt generated according to the profile of each subject. Afterward, we generated 10 images per prompt that were evaluated in the same way as human-created prompts. Table 1 shows the best scores obtained for each target image by both the subjects and the ChatGPT-4. As

it can be observed, for all cases, subject's prompts outperform the scores of ChatGPT, however it still shows a very competitive performance since the obtained scores are very similar to the prompts generated by humans. Interestingly, ChatGPT shows the most salient drop in the score rate when compared with subject 2, which prompts are the shorter and therefore the summary of them are even shorter provoking to lose important details for generating an effective prompt. We are planning to further explore how to better leverage the capabilities of LLMs for applying prompt engineering as an auxiliary tool for generating images. Due to the lack of space, the images with the highest score for each target with the summary prompt are shown in Figure 4 for subject 1 and in Figure 5 for the remaining subjects.

Table 1: Comparison of the best scores for synthetic images generated with Stable Diffusion using subject prompts versus the best score obtained with synthetic images generated with a prompt from ChatGPT-4 that combined all the prompts of a corresponding subject.

Target Image	Best Score					
	S1 Best	GPT-4 (S1)	S2 Best	GPT-4 (S2)	S3 Best	GPT-4 (S3)
1	89.84	80.03	84.08	64.99	84.77	83.84
2	82.86	80.91	83.35	79.00	83.99	79.30
3	84.08	80.71	80.27	75.93	81.20	77.39
4	79.00	73.49	77.69	64.21	79.30	66.85
5	86.28	86.04	89.84	84.52	90.62	90.33

lated to AI-generated art community.



Figure 5: Same comparison as in Figure 4, but without showing the prompts and scores. Note that the resemblance between the target and synthetic images is uncanny.

5 CONCLUSION

We have presented the preliminary findings of a methodology which utilises language and generative models such as CLIP, Stable Diffusion, and ChatGPT-4 to provide insights into the exploitation of a textual description in the context of parcel delivery conducted by an autonomous drone. Our research scenario envisages that the drone might reach the vicinity of a delivery point using GPS, but may still require assistance to pinpoint the exact delivery location. In the final stage of delivery, commonly referred to as the 'last-mile delivery', a delivery person often encounters difficulty in identifying the precise target location. Consequently, companies now frequently request customers to provide textual descriptions of the delivery location. Inspired by this scenario, we suggest that a delivery drone might also utilise these textual descriptions to generate synthetic images which reflect the delivery point, even if the drone has never visited the location previously.

Our initial results demonstrate that it is indeed possible to generate synthetic images that could serve as a guide to search for the target location using images captured by an on-board drone camera, given that such images can be compared

in a semantic shared space. We acknowledge that a more extensive testing phase involving more subjects and additional image data is needed. However, our preliminary findings are promising and indicate that synthetic generation with generative models such as Stable Diffusion model, can be leveraged with prompt engineering. Therefore, our future work will focus on automatic prompt generation given a single textual description. Indeed, we foresee that in terms of customer interaction, it may not be long before we are able to *speak to* a delivery drone to direct it to its destination, either for delivery or collection purposes.

ACKNOWLEDGEMENTS

We used ChatGPT-4 for prompt generation in the experiments. L.O.R.P. and A.A.C.P. are thankful to CONAHCYT in Mexico for their scholarships with numbers 924254 and 802791.

REFERENCES

- [1] Robin Kellermann, Tobias Biehle, and Liliann Fischer. Drones for parcel and passenger transportation: A literature review. *Transportation Research Interdisciplinary Perspectives*, 4:100088, 2020.

- [2] Nils Boysen, Stefan Fedtke, and Stefan Schwerdfeger. Last-mile delivery concepts: a survey from an operational research perspective. *Or Spectrum*, 43:1–58, 2021.
- [3] Xuping Wang, Linmin Zhan, Junhu Ruan, Jun Zhang, et al. How to choose “last mile” delivery modes for e-fulfillment. *Mathematical Problems in Engineering*, 2014, 2014.
- [4] Elżbieta Macioszek. First and last mile delivery—problems and issues. In *Advanced Solutions of Transport Systems for Growing Mobility: 14th Scientific and Technical Conference “Transport Systems. Theory & Practice 2017” Selected Papers*, pages 147–154. Springer, 2018.
- [5] Giuseppe Loianno, Chris Brunner, Gary McGrath, and Vijay Kumar. Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and IMU. *IEEE Robotics and Automation Letters*, 2(2):404–411, 2016.
- [6] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models, 2021.
- [7] Jose Martinez-Carranza, Richard Bostock, Simon Willcox, Ian Cowling, and Walterio Mayol-Cuevas. Indoor mav auto-retrieval using fast 6d relocalisation. *Advanced Robotics*, 30(2):119–130, 2016.
- [8] L Oyuki Rojas-Perez and Jose Martinez-Carranza. DeepPilot4pose: a fast pose localisation for mav indoor flight using the oak-d camera. *Journal of Real-Time Image Processing*, 20(1):8, 2023.
- [9] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
- [10] Vinicius Luis Trevisan de Souza, Bruno Augusto Dorta Marques, Harlen Costa Batagelo, and João Paulo Gois. A review on generative adversarial networks for image generation. *Computers Graphics*, 114:13–25, 2023.
- [11] Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, and Jie Tang. CogView: Mastering Text-to-Image Generation via Transformers. In *Advances in Neural Information Processing Systems*, volume 34, pages 19822–19835. Curran Associates, Inc., 2021.
- [12] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.
- [13] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. VisualBERT: A Simple and Performant Baseline for Vision and Language, 2019.
- [14] Tobias Hinz, Stefan Heinrich, and Stefan Wermter. Semantic Object Accuracy for Generative Text-to-Image Synthesis. *CoRR*, abs/1910.13321, 2019.
- [15] Vitali Petsiuk, Alexander E. Siemenn, Saisamrit Surbhera, Zad Chin, Keith Tyser, Gregory Hunter, Arvind Raghavan, Yann Hicke, Bryan A. Plummer, Ori Kerret, Tonio Buonassisi, Kate Saenko, Armando Solar-Lezama, and Iddo Drori. Human Evaluation of Text-to-Image Models on a Multi-Task Benchmark, 2022.
- [16] Lanxiao Wang, Wenzhe Hu, Heqian Qiu, Chao Shang, Taijin Zhao, Benliu Qiu, King Ngi Ngan, and Hongliang Li. A survey of vision and language related multi-modal task. *CAAI Artificial Intelligence Research*, 1(2):111–136, 2022.
- [17] Jing Gu, Eliana Stefani, Qi Wu, Jesse Thomason, and Xin Wang. Vision-and-language navigation: A survey of tasks, methods, and future directions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7606–7623, Dublin, Ireland, May 2022.
- [18] Renrui Zhang, Ziyao Zeng, Ziyu Guo, and Yafeng Li. Can Language Understand Depth? In *Proceedings of the 30th ACM International Conference on Multimedia*, MM ’22, page 6868–6874, New York, NY, USA, 2022. Association for Computing Machinery.
- [19] Jose Martinez-Carranza, Delia Hernández-Farías, Leticia Oyuki Rojas Pérez, and Aldrich Cabrera Ponce. Language meets yolov8 for metric monocular slam. *Journal of Real-Time Image Processing*, 20, 05 2023.
- [20] Ivan Kapelyukh, Vitalis Vosylius, and Edward Johns. DALL-E-Bot: Introducing Web-Scale Diffusion Models to Robotics. *IEEE Robotics and Automation Letters*, 8(7):3956–3963, jul 2023.
- [21] OpenAI. ChatGPT [Large language model], 2023.
- [22] Vivian Liu and Lydia B Chilton. Design guidelines for prompt engineering text-to-image generative models. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI ’22, 2022.
- [23] Jonas Oppenlaender. A Taxonomy of Prompt Modifiers for Text-To-Image Generation, 2022.
- [24] Anna Notaro. State of the Art: A.I. through the (artificial) artist’s eye. In *EVA*, 2020.

A Lightweight Learning-based Visual-Inertial Odometry

Yingfu Xu and Guido C. H. E. de Croon *
Delft University of Technology

ABSTRACT

In this paper, we propose a learning-based lightweight visual-inertial odometry (VIO) based on an uncertainty-aware pose network and an extended Kalman filter (EKF). The pose network serving as the VIO vision front-end predicts the relative motion of the camera between consecutive image frames and estimates the prediction uncertainty. The training of the pose network can be conducted without requiring ground-truth labels. The distributions of visual measurements are fused with inertial measurements by an EKF that is the VIO back-end. Evaluations show that the proposed VIO fails to outperform a state-of-the-art feature-point-based VIO solution in accuracy. But it has high time efficiency, translational motion estimation with metric scale, estimation of gravity direction, and generalization to new environments. So, unlike most works on learning-based visual ego-motion estimation in the literature, the proposed VIO can be directly deployed on an MAV. The comparative studies of supervision signals and forms of translational motion prediction provide insights that can contribute to future research.

1 INTRODUCTION

Ego-motion estimation of micro air vehicles (MAVs) is essential for autonomous flight and has been a major research topic in the robotics domain. Monocular cameras are often used for ego-motion estimation of lightweight MAVs in environments without stable GPS signals, because of their small size, lightweight, low energy consumption, and rich environmental information captured. After years of research, ego-motion estimation approaches that use vision in conjunction with an inertial measurement unit (IMU), *i.e.*, visual-inertial odometry (VIO), are widely applied to MAVs. Many mainstream VIO approaches, *e.g.* multi-state constraint Kalman filter (MSCKF) [1], use visual feature points as the vision processing front-end. Assuming the scene is stationary, information about the camera motion can be obtained by tracking the two-dimensional (2-d) pixel locations of the same feature point in multiple temporally consecutive images. Therefore, extracting a sufficient number of feature points and then tracking or matching them accurately across frames is the key

to accurate ego-motion estimation. However, because of the dependence on appearance consistency and image gradient, visual feature points are susceptible to varying illumination and image blur.

As an alternative to feature points, researchers have been exploring deep learning for visual ego-motion estimation. Learning-based approaches train one or multiple artificial neural network(s) (ANN) to predict the relative pose [2, 3] between temporally consecutive images or visual correspondences that encode ego-motion, *e.g.*, optical flow [4, 5] and planar homography transformation [6]. A pose network can be trained by the ground-truth relative pose in supervised learning. But the acquisition of accurate camera position and orientation in the real world often requires external sensors such as motion capture systems. So the scenes of captured images for training are restricted to the range of the stationary motion capture sensors. Thus it limits the size of the training dataset and deteriorates the generalization capacity of the pose network. In contrast, self-supervised learning is an attractive approach for relieving the need for ground truth. The first self-supervised learning scheme of a pose network was proposed in [2], where a monocular depth network is trained simultaneously. It is sometimes referred to as self-supervised structure-from-motion (SfM) in the literature. According to the predicted dense depth and relative camera pose, a virtual view can be synthesized by reprojecting the world points into the image frame. The photometric error between the synthesized view and the actually captured image is the main supervision signal. This scheme has been further developed by many follow-up works and is adopted in this paper. A more detailed description is provided in Section 2.

It has been observed that ANNs show better robustness towards unfavorable conditions such as illumination change, motion blur, and dynamic scenes [3, 5]. There are learning-based approaches [4, 5] achieving better accuracy on the EuRoC MAV datasets [7] than a traditional approach [8]. But general challenges exist in learning-based approaches. Firstly, many works train and test on the same dataset without considering the network's generalization capability. Secondly, achieving high accuracy requires a significant amount of computational power, making such approaches unsuitable for computationally constrained MAVs. Thirdly, many works cannot be directly deployed on MAVs due to various limitations. For example, translational motion lacks scale in [5]. The scale is ambiguous in the case of self-supervised learning with monocular videos [2, 9, 10]. The scale is also unknown in the case of self-supervised VIO that processes IMU measurements with an ANN [11]. Because the loss function

*Email addresses: yingfu.xu.94@gmail.com, g.c.h.e.decroon@tudelft.nl

has no constraint on the scale, the metric-scale accelerometer measurements can be scaled arbitrarily by the ANN. Another problem in using an IMU network is that, when the supervision signal only regularizes the relative pose [11], the gravity information encoded in the accelerometer measurements is not exploited. So the gravity direction remains unknown, though it is observable for a monocular visual-inertial system [12].

This paper aims to address the above-mentioned challenges by introducing a learning-based VIO. It can be directly deployed on an MAV due to its notable characteristics, such as high efficiency, translational motion estimation with metric scale, estimating the direction of gravity, and generalizing to new environments. The vision front-end is a pose network that infers the relative rotation and translation of the camera from an image pair. It is capable of estimating the prediction uncertainty, as is useful for the EKF-based VIO back-end. In addition, the network's training can be conducted with or without ground-truth labels. In the latter case, self-supervised learning schemes for pose and depth [9, 10] are adopted to first train "teacher" networks with high prediction accuracy. The pose predictions of the teacher networks serve as the learning target of the lightweight "student" pose network that is the VIO vision front-end. The effects of three elements of the pose network design are evaluated and analyzed. They are the supervision signal (with vs. without ground truth), the scale of translational motion prediction (with metric scale vs. direction only), and with vs. without fine-tuning on the target dataset.

2 TEACHER NETWORKS

2.1 Improved Self-Supervised SfM

The mission of the teacher networks is to perform self-supervised learning and provide accurate relative pose predictions that serve as the learning targets in the training of a lightweight pose network. We take Monodepth2 [9] and its open-sourced code as the base of the teacher networks and adopt two additional methods to achieve higher pose prediction accuracy. The first is a loss term L_{geo} that constrains the depth predictions, proposed by [10]. It is based on the geometry consistency of the 3-d positions of the world points. The second is to conduct the pose network inference multiple times by iterative view synthesis that relies on the depth map prediction, proposed by [13]. Instead of predicting the total transformation by a single inference, the pose network incrementally refines the relative pose prediction by inferring from input images that are more and more similar.

The self-supervised learning loss function of the teacher networks is shown in Eq. (1). V denotes the set of all pixels. L_{photo} is referred to as the reprojection-based loss in some works and in this paper. L_{photo} combines the $L1$ loss of pixel-wise photometric error and the structured similarity index measure (SSIM) loss, with $\alpha = 0.85$. N is the total number of iterations of pose network inference. The geome-

try consistency loss L_{geo} is computed only once using the final pose prediction after all iterations. Hyperparameters λ_{geo} and λ_i are scalar weights of loss terms. $\mathcal{M}_{\text{auto}}(\cdot)$ stands for applying the auto-masking strategies of Monodepth2 to the pixels. The smoothness loss for depth prediction is the edge-aware smoothness implemented by the open-sourced code of Monodepth2. We omit its expression in Eq. (1). An interested reader can refer to [9, 10, 13] for details of the loss function.

$$L = \lambda_{\text{geo}} \cdot L_{\text{geo}} + \sum_{i=1}^N \lambda_i \cdot L_{\text{photo},i}$$

$$L_{\text{photo},i} = \frac{1}{|V|} \sum_{k \in V} \mathcal{M}_{\text{auto}} \left(\frac{\alpha}{2} (1 - \text{SSIM}(I_t, \tilde{I}_{s,i})_k) \right. \\ \left. + (1 - \alpha) \cdot |I_{t,k} - \tilde{I}_{s,i,k}| \right) \quad (1)$$

Instead of predicting depth maps at four scales as Monodepth2 does, we predict a single depth map at the full resolution. It results in higher accuracy in both depth and pose predictions. To better learn how to handle fast motion, our implementation uses a bigger temporal step in loading image snippets. There is 25% possibility to load I_{i-2} , I_i , and I_{i+2} and 75% possibility to load the neighboring images I_{i-1} , I_i , and I_{i+1} .

2.2 Datasets and Network Training

TartanVO [5] and Droid-SLAM [4] are two of a few works that achieve good performance of cross-dataset generalization. Both of them use TartanAir [14] dataset for training. We take most of the images (296,899) for training and a small number (1,522) for in-domain testing. Testing images are randomly sampled from all the sequences.

A pose network infers relative pose from raw images, so camera intrinsics affect the prediction. Such a pose network requires the input image to always have the specific image resolution and camera intrinsics. Therefore, in cross-dataset testing and real-world deployment, the images need to be pre-processed to be the same as training images in terms of resolution and camera intrinsics. In this paper, input images to a pose network are required to be grey-scale and have a resolution of 192×352 and intrinsics $f_x = 176$, $f_y = 176$, $c_x = 176$, $c_y = 96$. This setting fits both the training set (TartanAir) and the testing set (EuRoC MAV dataset [7]). Images of TartanAir lose pixels near the top and bottom edges after being transformed to this setting. But if we transform the images of EuRoC to have the same resolution and camera intrinsics as TartanAir, the characteristics of the lens and optic sensor of the camera used in collecting the EuRoC dataset lead the images to have black curving edges. This phenomenon reveals the lack of cross-camera generalization of pose networks. A solution based on preprocessing the input optical flow map is proposed in [14], while the pose network in this paper uses raw images as input. So the cross-camera generalization is not solved.

The networks in this section are trained on the TartanAir training set for 100 epochs. We utilized the 1 cycle learning rate policy [15] with the maximum learning rate of $2.5\text{e-}4$ in self-supervised learning. The architectures of the pose and depth networks are based on ResNet-18, the same as Monodepth2. The number of iterations $N = 3$. The weights of the loss for each iteration are $\lambda_0 = 0.25$, $\lambda_1 = 0.5$, and $\lambda_2 = 1.0$. The weight for geometry consistency $\lambda_{\text{geo}} = 0.2$. The batch size is 32.

We compare the pose prediction accuracy of the iterative teacher networks trained by the loss function (Eq. (1)) with the original Monodepth2 in Table 1. The accuracy metric is the average of the norms of error vectors, $e = \frac{1}{N} \sum \|\mathbf{v}_{\text{GT}} - \mathbf{v}_{\text{Pred.}}\|_2$. The rotation is expressed by axis-angle and implemented in the open-sourced code of Monodepth2. The predicted translation and the ground-truth translation are normalized to 3-d unit vectors before calculating the error because the scale of the translation prediction is ambiguous. When the translation predictions are used as the learning targets of the uncertainty-aware pose network, they are also normalized.

Network	TartanAir Rot.	TartanAir Trans.	EuRoC Rot.	EuRoC Trans.
ours	1.99e-03	8.77e-02	3.09e-03	2.39e-01
ours-FT	-	-	1.73e-03	1.62e-01
Monodepth2	2.21e-03	1.23e-01	4.08e-03	2.89e-01
Monodepth2- FT	-	-	3.18e-03	2.48e-01

Table 1: Average pose prediction errors on TartanAir testing set and EuRoC testing set. “FT” indicates that the networks are fine-tuned on the EuRoC training set based on the parameters trained on the TartanAir training set. **Bold** represents the best.

As shown in Table 1, we perform an in-domain test on the TartanAir testing set and an out-of-domain test on a part of the EuRoC MAV dataset [7]. There are 11 sequences in the EuRoC dataset. The EuRoC training set is made of all sequences except for *V103*, *V203*, and *MH05*. These three sequences are used for testing. From the first and third rows of Table 1, we can see that our networks have smaller average errors in pose predictions on both tests.

We finetune the networks trained on the TartanAir dataset on the training set of EuRoC and test them on the three testing sequences, as shown in the second and fourth rows of Table 1. Although TartanAir is a large-size dataset with a wide distribution, we still observe that fine-tuning brings obvious improvement in pose prediction accuracy.

3 VIO BASED ON POSE NETWORK AND EKF

In the previous section, we introduce the methodology to train the self-supervised teacher network. As already explained in the introduction, the teacher networks are not suit-

able for the ego-motion estimation of an MAV. Specifically, the pose prediction of the teacher networks relies on one forward pass of the depth network and multiple forward passes of the pose network, which require considerable computational power. In addition, the translational predictions have an ambiguous scale and the gravity direction is not estimated.

In this section, we introduce a computationally efficient VIO that can estimate metric-scale translational motion and the gravity direction. Its vision front-end is a pose network. It performs a single forward pass for a pose prediction. The back-end is a simple EKF. We train several pose networks using two supervision signals and compare the resulting VIO accuracy. They are, respectively, the ground-truth poses and the teacher networks’ pose predictions. When using the pose predictions of teacher networks for training, the front-end pose network is a student network, and the whole learning pipeline does not require ground-truth labels of the pose. Different from the teacher networks, the front-end pose network not only predicts the relative pose but also estimates the uncertainty of its prediction. So it is called an uncertainty-aware pose network.

3.1 Uncertainty-Aware Pose Network

The architecture of the pose network is based on the pose network used by Monodepth2. It has an encoder based on ResNet-18. We modify the architecture of the decoder part to enable it for uncertainty estimation. The output tensor of the last convolutional layer is the input to two fully connected (FC) subnetworks. One FC network predicts the mean value, and the other predicts the *aleatoric* uncertainty that reflects the noise inherent in the network input. It is referred to as predictive uncertainty in [6] and this paper. The two FC networks have the same architecture of two FC layers. There is a 5% dropout for the input of each FC layer, to estimate the *epistemic* uncertainty using MC-Dropout [16]. *Epistemic* uncertainty captures the ignorance about the ideal network that maps noiseless input to the desired output. It is called empirical uncertainty in [6] and this paper. The total uncertainty is the sum of predictive uncertainty and empirical uncertainty. The uncertainty estimation of the front-end uncertainty-aware pose network follows the same methodologies as in [6]. An interested reader can read [6] for more details. We introduce the most important steps in the following.

Treating the observed pose as a sample from a Gaussian distribution, the loss function for learning predictive uncertainty is the negative log-likelihood (NLL) loss, as shown in Eq. (2). This loss function is used for the same purpose in [17].

$$L_{\text{NLL}} = \sum_{n=1}^6 \frac{1}{2\sigma_{n,\text{pred.}}^2} \|\mathbf{T}_n - \mu_n\|^2 + \frac{1}{2} \log(\sigma_{n,\text{pred.}}^2) \quad (2)$$

μ_n is the prediction of the mean value of the relative pose vector. $\sigma_{n,\text{pred.}}^2$ is the variance prediction corresponding to the predictive uncertainty. n indexes over the six dimensions of

relative pose (3-d rotation and 3-d translation). T_n denotes the learning target of the mean prediction μ_n . T_n can be the ground-truth relative pose or the pose prediction of the trained teacher networks. The pose network can be trained to predict translational motion with the metric scale. It can also be trained to predict normalized translation expressed by a unit vector that indicates the motion direction without scale. We implement both of them and compare them in Subsection 3.3. When the pose network is trained to predict translational direction, its prediction of the mean translation is normalized. The translational part of T_n is normalized accordingly.

MC-Dropout [16] requires multiple forward passes to sample from the distributions of network parameters. m indexes over the forward passes. We set the number of MC-Dropout sampling $M=8$. After each forward pass, a mean prediction $\mu_{n,m}$ is obtained, as well as a variance prediction of predictive uncertainty $\sigma_{\text{pred},n,m}^2$. As Eq. (3) shows, the variance of empirical uncertainty $\sigma_{\text{emp},n}^2$ is calculated empirically from the multiple mean predictions $\mu_{n,m}$. The total variance σ_n^2 is the sum of the empirical variance $\sigma_{\text{emp},n}^2$ and the average of predictive variances $\sigma_{\text{Avg}, \text{pred}, n}^2$.

$$\begin{aligned}\sigma_n^2 &= \sigma_{\text{Avg}, \text{pred}, n}^2 + \sigma_{\text{emp}, n}^2 \\ \sigma_{\text{Avg}, \text{pred}, n}^2 &= \frac{1}{M} \sum_{m=1}^M \sigma_{\text{pred}, n, m}^2 \\ \sigma_{\text{emp}, n}^2 &= \frac{1}{M} \sum_{m=1}^M (\mu_{n,m} - \mu_n)^2, \quad \mu_n = \frac{1}{M} \sum_{m=1}^M \mu_{n,m}\end{aligned}\quad (3)$$

3.2 EKF-based Back-end

The VIO back-end is an EKF. It is a simplified variant of the EKF-based back-end of the robocentric VIO [18]. The robocentric VIO keeps a window of the previous camera poses in the state vector. Our simplified variant only estimates the relative pose between the current IMU frame and the local frame of reference. The EKF state vector is defined as

$$\mathbf{x} := \begin{bmatrix} {}^{R_k} \mathbf{p}_G, & {}^G \mathbf{q}, & \mathbf{g}_{R_k}, \\ {}^{I_t} \mathbf{p}_{R_k}, & {}^{I_t} \mathbf{q}, & \mathbf{v}_{I_t}, \quad \mathbf{b}_a, \quad \mathbf{b}_g \end{bmatrix}. \quad (4)$$

I_t is the current IMU frame at time t . When a new image has been captured, the new reference frame R is set to be the same as the I_t at the time. R_k is the current reference frame. It is the k th reference frame since the VIO is initialized. G stands for the global frame. It is the first reference frame R_0 , i.e., the IMU frame when the first image is captured after the initialization of the VIO. ${}^{R_k} \mathbf{p}_G$ is a translation vector pointing from the origin of G to the origin of R_k , expressed in R_k . It is about the global position of R_k . ${}^{I_t} \mathbf{p}_{R_k}$ is a translation vector pointing from the origin of R_k to the origin of I_t , expressed in I_t . It is about the local position of I_t relative to R_k . ${}^G \mathbf{q}$ is the Hamilton quaternion reflecting the relative rotation between G and R_k . ${}^{I_t} \mathbf{q}$ reflects the relative rotation between R_k and

I_t . $\mathbf{g}_{R_k}$ indicates the gravity vector expressed in R_k . $\mathbf{v}_{I_t}$ is the translational velocity of the IMU expressed in I_t . $\mathbf{b}_a$ and $\mathbf{b}_g$ are respectively the additive bias on accelerometer and gyroscope.

As shown in Eq. (5), IMU measurements are modeled as the sum of the desired actual value ($\hat{\mathbf{a}}$ and $\hat{\boldsymbol{\omega}}$), additive bias, and white Gaussian noise ($\mathbf{w}_a$ and $\mathbf{w}_g$).

$$\mathbf{a}_m = \hat{\mathbf{a}} + \mathbf{b}_a + \mathbf{w}_a, \quad \boldsymbol{\omega}_m = \hat{\boldsymbol{\omega}} + \mathbf{b}_g + \mathbf{w}_g \quad (5)$$

$$\begin{aligned}{}^{I_t} \dot{\mathbf{p}}_{R_k} &= -[\hat{\boldsymbol{\omega}}]_{\times} \cdot {}^{I_t} \mathbf{p}_{R_k} + \mathbf{v}_{I_t} + \mathbf{w}_p, \\ \dot{\mathbf{v}}_{I_t} &= -[\hat{\boldsymbol{\omega}}]_{\times} \cdot \mathbf{v}_{I_t} + \hat{\mathbf{a}} + \mathbf{R}({}^{I_t} \mathbf{q})^T \cdot \mathbf{g}_{R_k}, \\ {}^{I_t} \dot{\mathbf{q}} &= \frac{1}{2} {}^{I_t} \mathbf{q} \otimes \begin{bmatrix} 0 \\ \hat{\boldsymbol{\omega}} \end{bmatrix}, \\ \dot{\mathbf{b}}_a &= \mathbf{w}_{b_a}, \quad \dot{\mathbf{b}}_g = \mathbf{w}_{b_g}.\end{aligned}\quad (6)$$

Eq. (6) shows the IMU-driven state dynamics ($\dot{\mathbf{x}}$). $[\hat{\boldsymbol{\omega}}]_{\times}$ represents the skew-symmetric matrix associated with $\hat{\boldsymbol{\omega}}$. $\mathbf{w}_p$ is the process noise in position integration. $\mathbf{R}({}^{I_t} \mathbf{q})$ is a transformation function from ${}^{I_t} \mathbf{q}$ to $SO(3)$ rotation matrix that maps a vector expressed in I_t to its expression in R_k . $\otimes$ denotes quaternion product. We utilize the techniques introduced in [19] for quaternion-related calculation.

The filter states in Eq. (6) are propagated by the IMU measurements until a new image I_{k+1} is captured. The pair of I_{k+1} and the previous image I_k are the inputs of the uncertainty-aware pose network that is introduced in Subsection 3.1. The network outputs $\boldsymbol{\mu}_t$, $\boldsymbol{\sigma}_t^2$, $\boldsymbol{\mu}_\theta$, and $\boldsymbol{\sigma}_\theta^2$. They are the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}$ of translation $\mathbf{t}$ and rotation $\boldsymbol{\theta}$, respectively.

$$\begin{aligned}\mathbf{z}_{\boldsymbol{\mu}_t} &= \mathbf{R}_I^C \cdot (\mathbf{t}_{IC} + {}^{I_t} \mathbf{p}_{R_k} - \mathbf{R}({}^{I_t} \mathbf{q})^T \cdot \mathbf{t}_{IC}) + \mathbf{w}_t, \\ \mathbf{z}_{\boldsymbol{\mu}_\theta} &= \mathbf{R}_I^C \cdot \boldsymbol{\theta}({}^{I_t} \mathbf{q}) + \mathbf{w}_\theta.\end{aligned}\quad (7)$$

The measurement equations are shown in Eq. (7). $\mathbf{R}_I^C$ is the rotation matrix from the IMU frame to the camera frame. $\mathbf{t}_{IC}$ is the translation vector points from the IMU to the camera, expressed in the IMU frame. $\boldsymbol{\theta}(\cdot)$ converts a quaternion to an axis-angle expression. Elements of $\boldsymbol{\sigma}_t^2$ and $\boldsymbol{\sigma}_\theta^2$ are used as the diagonal elements of the measurement noise matrix. After the measurement update, the *a posteriori* relative pose estimation ${}^{I_t} \hat{\mathbf{p}}_{R_k}$ and ${}^{I_t} \hat{\mathbf{q}}$ is composed to the global pose, as shown in Eq. (8).

$$\begin{aligned}{}^{R_{k+1}} \mathbf{q} &= {}^{I_t} \hat{\mathbf{q}} \otimes {}^{R_k} \mathbf{q}, \\ {}^{R_{k+1}} \mathbf{p}_G &= \mathbf{R}({}^{I_t} \hat{\mathbf{q}})^T \cdot {}^{R_k} \mathbf{p}_G + {}^{I_t} \hat{\mathbf{p}}_{R_k}\end{aligned}\quad (8)$$

The current IMU frame I_t becomes the new reference frame R_{k+1} . The expression of the gravity vector $\mathbf{g}_{R_{k+1}}$ in R_{k+1} is calculated as $\mathbf{g}_{R_{k+1}} = \mathbf{R}({}^{I_t} \hat{\mathbf{q}})^T \cdot \hat{\mathbf{g}}_{R_k}$. ${}^{I_t} \mathbf{p}_{R_{k+1}}$ and ${}^{I_t} \mathbf{q}$ are set to a zero vector and a unit quaternion whose vector part is a zero vector, respectively. Their corresponding elements in the covariance matrix are set to zeros too. Readers who

Table 2: Accuracy of the proposed method (top group, rows 1 to 6) compared with the baseline method (bottom group, rows 7 and 8). *V101* to *MH05* shown in the eleven columns from the right are the names of flight sequences of the EuRoC MAV dataset. The data below the sequence names show the root-mean-square errors (RMSE) of the absolute translation errors (ATEs) of the estimated trajectories. **Bold** represents the overall best and underline represents the best of the proposed method.

ID	FT ¹	GT ²	MS ³	V101	V102	V103	V201	V202	V203	MH01	MH02	MH03	MH04	MH05
①				4.24	<u>2.34</u>	3.04	4.05	4.97	5.88	6.42	<u>3.92</u>	4.79	14.1	4.04
②	✓			-	-	2.23	-	-	4.33	-	-	-	-	4.25
③		✓		3.48	2.51	2.74	4.07	5.49	7.73	<u>5.60</u>	3.93	6.18	17.7	19.8
④	✓	✓		-	-	2.42	-	-	7.99	-	-	-	-	8.16
⑤		✓	✓	<u>1.23</u>	2.86	3.74	<u>1.86</u>	<u>4.37</u>	<u>4.27</u>	7.24	5.06	4.26	<u>3.24</u>	3.68
⑥	✓	✓	✓	-	-	<u>1.55</u>	-	-	4.36	-	-	-	-	<u>3.04</u>
⑦	MSCKF (51 pts)			0.16	0.13	0.12	245	0.13	0.16	38.0	5.20	130	2.35	1.00
⑧	MSCKF (199 pts)			0.09	0.09	0.11	0.12	0.10	0.20	0.34	0.24	58.5	0.65	1.54

¹ Fine-tuning (FT) the pose network on the training sequences of the EuRoC dataset.

² Using ground-truth (GT) labels as the learning targets in the NLL loss. If not using GT, the learning targets are the predictions of the self-supervised teacher networks.

³ The pose network learns to predict translational motion with the metric scale (MS). Otherwise, the prediction is the normalized translation, a unit vector.

are interested in the filter design can refer to [18] for more details.

For the EKF, when μ_t is the normalized translation, the propagated camera translation is normalized accordingly. The measurement equation of translational motion is then

$$z_{\mu_t} = R_I^C \cdot \frac{t_{IC} + {}^{I_t}p_{R_k} - R_{(R_k)}^{(I_t)} q)^T \cdot t_{IC}}{\|t_{IC} + {}^{I_t}p_{R_k} - R_{(R_k)}^{(I_t)} q)^T \cdot t_{IC}\|} + w_t \quad (9)$$

3.3 Evaluation

In the previous part of this section, we have successively introduced the vision front-end which is the uncertainty-aware pose network and the EKF-based robocentric back-end. The VIO system consisting of them is referred to as PoseNet-VIO in this paper. In this subsection, PoseNet-VIO is evaluated on the EuRoC MAV dataset [7]. The PoseNet-VIO variants in Table 2 have the same parameters in the EKF-based back-end. Their vision front-ends, the pose networks, are different. They are trained by the NLL loss (Eq. (2)) with different learning targets. The learning targets of ① and ② are the pose predictions of the self-supervised teacher networks. The remaining four networks learn from the ground truth. The translation prediction of ① to ④ in Table 2 are normalized to a unit vector that indicates the direction of translational motion. In contrast, ⑤ and ⑥ predict translational motion with the metric scale that is learned from the ground-truth labels of the relative poses. Since we use monocular videos to train the teacher networks, their prediction of translational motion has an ambiguous scale. Therefore, the pose networks trained by the teacher networks can only have normalized translational motion predictions. ②, ④, and ⑥ are fine-tuned on the EuRoC training datasets based on ①, ③, and ⑤, respectively. The teacher networks

of ② are fine-tuned on the EuRoC training set based on the parameters of the teacher networks of ①. PoseNet-VIO with a fine-tuned network is evaluated on the EuRoC testing sequences. All the front-end pose networks are trained on the TartanAir dataset for 100 epochs. Fine-tuning on the EuRoC training set lasts for 50 epochs.

The root-mean-square error (RMSE) of absolute translation errors (ATE) is the metric of VIO accuracy. The alignment of the estimated trajectory and the ground-truth trajectory has 4-DoF (yaw and 3-d translation). The calculation is conducted by [20]. PoseNet-VIO is compared with a state-of-the-art (SOTA) VIO MSCKF [1] implemented by [21]. The C++ code of PoseNet-VIO is implemented based on the open-sourced code of [21]. Here we only compare with one SOTA VIO since the gap in accuracy is big, as shown in Table 2. The comparison of MSCKF with other VIO solutions can be found in [22].

At the beginning of the five EuRoC sequences collected in the machine hall, the MAV was moved by a human operator and then stayed stationary for a while before taking off. For a PoseNet-VIO whose front-end pose network has normalized translation predictions, the lack of translational motion leads to drift. So, when testing PoseNet-VIO variants ① to ④, the starting time points of the *Machine Hall (MH)* sequences are about one second before the takeoff. For MSCKF, if there was a significant drift in the beginning, we did the same.

A design target of PoseNet-VIO is high efficiency. Thus it has non-iterative network inference and a basic EKF. The time consumption is constant. But since MSCKF uses feature points, the number of points affects the time consumption. The time consumptions of PoseNet-VIO and MSCKF are measured during the tests on *MH05* sequence of EuRoC. For PoseNet-VIO, the average total time cost is 7.811 ms, of

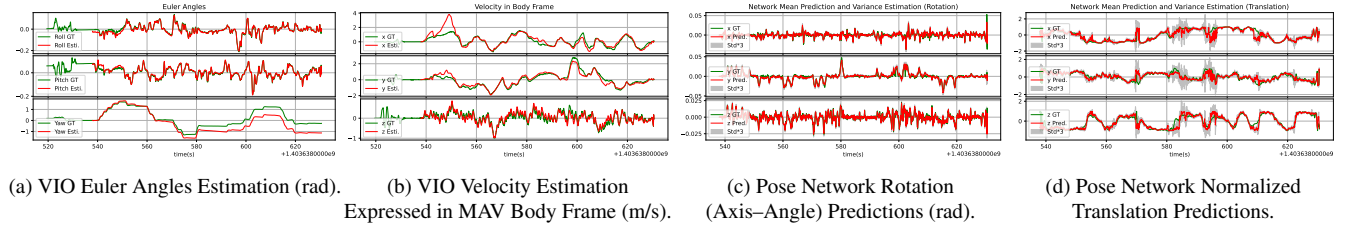


Figure 1: Results of PoseNet-VIO variant ①, tested on *MH05*. The two subplots on the left show the EKF *a posteriori* estimations (VIO outputs) of the MAV attitude and translational velocity. The attitude is relative to the global frame whose *z*-axis is opposite to the gravity direction. The two subplots on the right show the outputs (mean prediction and uncertainty estimation of the relative pose) of the front-end pose network.

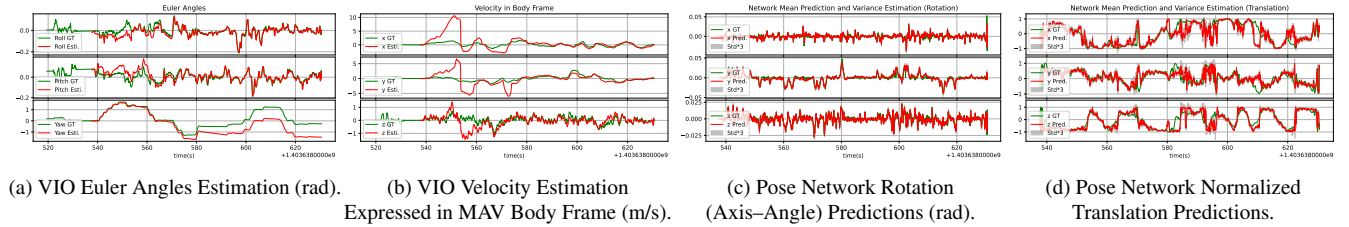


Figure 2: Results of PoseNet-VIO variant ③, tested on *MH05*.

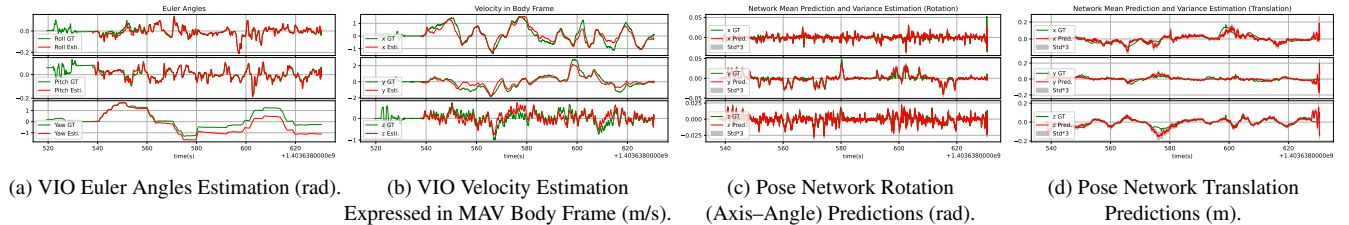


Figure 3: Results of PoseNet-VIO variant ⑤, tested on *MH05*.

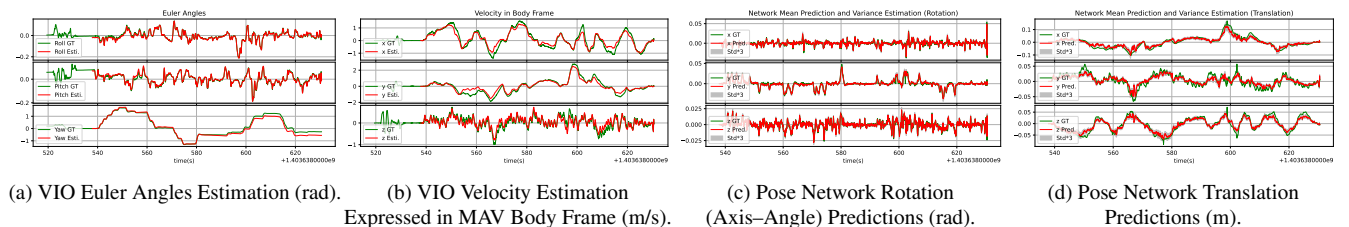


Figure 4: Results of PoseNet-VIO variant ⑥, tested on *MH05*.

which 6.036 ms is for network inference. For the default setting of MSCKF (⑧ in Table 2), the average total time cost is 17.415 ms, with 7.583 ms for processing feature points. The average number of points is around 199. We modified the number of feature points to track to a smaller number than the default to make the average of the total time consumption of processing a frame close to the time consumption of PoseNet-VIO. When the average number of maintained points in an image frame is around 51 (⑦ in Table 2), the average total time cost is 7.933 ms, with 3.821 ms for feature point tracking and detection.

In the sequences *V201*, *MH01*, and *MH03*, MSCKF (⑦) drifts after very slow motion, which leads to big trajectory errors. But in general, as shown in Table 2, PoseNet-VIO is far less accurate than MSCKF. Comparing the variants of PoseNet-VIO, a noticeable phenomenon is that networks predicting translational motion with metric scale lead to better VIO accuracy than networks predicting normalized translation. Comparing the ATEs of ③ and ⑤, we can see that ⑤ has advantages in seven out of the eleven sequences. This is further evidenced by Fig. 2 (b) and Fig. 3 (b), where the estimated velocity of ⑤ is more accurate. Without utilizing the knowledge of objects' sizes in the scene, the metric scale is unobservable for a monocular video. The metric-scale translation prediction is based on the knowledge of the training set. It can be problematic when generalizing to a different dataset. But Fig. 3 (d) shows that the errors of translational motion predictions are acceptable. The generalization is fine. We attribute it to the large number and wide distribution of the training samples. Subplot (b) for VIO velocity estimations of Fig. 2 shows that using normalized translation (③) leads to big drifts in the beginning before the estimations converge after around 35 seconds. For ①, it takes around 10 seconds for the velocity estimation to converge, as shown in subplot (b) of Fig. 1. While using metric-scale translation does not have this problem, as shown in the two subplots (b) of Fig. 3 and Fig. 4. In summary, given the current design, using normalized translation leads to worse VIO accuracy in cross-dataset evaluation.

Another phenomenon worth noticing is the effect of fine-tuning. Fine-tuning brings better accuracy for all the variants, indicating that although we have a big-scale and widely distributed training set, the generalization capacity can still be improved. When the translation predictions have the metric scale, we can see better accuracy in all subplots of Fig. 4 than Fig. 3. Especially for the pose network's predictions of translational motion shown in subplot (d), the predictions are more accurate after fine-tuning. The possible reason is that the network further learns the metric scale of the objects captured in the EuRoC training set.

PoseNet-VIO variant ① interests us most because the training does not use ground-truth labels or in-domain data. Given the not-excellent but acceptable accuracy in Euler angle estimation and velocity estimation after the convergence,

as shown in Fig. 1, we think this PoseNet-VIO variant can act as an attitude and velocity estimator and be used for short-term navigation. From Table 2, Fig. 1, and Fig. 2, we notice that using the ground-truth poses in the training of the front-end pose network (③) does not improve the VIO accuracy over using the predictions of the self-supervised teacher networks (①). In eight out of eleven sequences, PoseNet-VIO variant ① has better accuracy. The accuracy of network predictions of rotation has no clear difference, as shown in the two subplots (c) of Fig. 1 and Fig. 2. But for the translation prediction shown in the two subplots (d), ① performs better than ③, especially in terms of uncertainty estimation. The three-time standard deviations of ① better reflect the errors of the mean predictions. This phenomenon indicates that self-supervised teacher networks are powerful replacements for ground-truth labels.

As shown in the subplot (c) of Fig. 1 to 4, the network predictions of relative rotation are well aligned with the ground truth. The translation predictions are relatively worse and noisier. Around 585, 595, and 610 seconds, translation predictions are obviously inaccurate, as shown in subplot (d) of Fig. 1. From subplot (b), we can see that the velocity is slow during those time slots. Slow velocity is a possible trigger of inaccurate translation prediction. More research is required to enable the network that predicts translation direction to better cope with slow motion. About potential ways of improving the PoseNet-VIO, one way is a VIO back-end that utilizes network predictions of multiple time steps instead of only the newest network prediction (current design). Another way is to learn the metric scale from the self-supervised teacher networks. It requires the teacher networks to be trained on stereo videos.

4 CONCLUSIONS

In this paper, we proposed the PoseNet-VIO based on an uncertainty-aware pose network and an EKF. The accuracy of PoseNet-VIO is worse than mainstream VIO solutions, but it may act as an efficient attitude and velocity estimator for short-term navigation. Based on the cross-dataset evaluation and the comparison between different types of supervision, we have the following findings. a) The simulation-to-reality generalization capacity of the pose network is generally satisfactory, also for the metric-scale translation prediction. b) Training an uncertainty-aware pose network using the predictions of self-supervised teacher networks leads to rivaling VIO accuracy to using ground-truth pose. c) Metric-scale translational motion predictions produce better VIO accuracy than normalized translational direction predictions. These findings can be valuable to future research on learning-based visual ego-motion estimation.

REFERENCES

- [1] Mingyang Li and Anastasios I Mourikis. High-precision, consistent ekf-based visual-inertial odome-

- try. *The International Journal of Robotics Research*, 32(6):690–711, 2013.
- [2] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1851–1858, 2017.
- [3] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks. *The International Journal of Robotics Research*, 37(4-5):513–542, 2018.
- [4] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Advances in neural information processing systems*, 34:16558–16569, 2021.
- [5] Wenshan Wang, Yaoyu Hu, and Sebastian Scherer. Tartanvo: A generalizable learning-based vo. In *Conference on Robot Learning*, pages 1761–1772. PMLR, 2021.
- [6] Yingfu Xu and Guido CHE de Croon. Cuahnvio: Content-and-uncertainty-aware homography network for visual-inertial odometry. *arXiv preprint arXiv:2208.13935*, 2022.
- [7] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016.
- [8] 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.
- [9] Clément Godard, Oisín Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3828–3838, 2019.
- [10] Jiawang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. Unsupervised scale-consistent depth and ego-motion learning from monocular video. *Advances in neural information processing systems*, 32, 2019.
- [11] Yasin Almalioglu, Mehmet Turan, Muhamad Risqi U Saputra, Pedro PB de Gusmão, Andrew Markham, and Niki Trigoni. Selfvio: Self-supervised deep monocular visual-inertial odometry and depth estimation. *Neural Networks*, 150:119–136, 2022.
- [12] Guoquan Huang. Visual-inertial navigation: A concise review. In *2019 international conference on robotics and automation (ICRA)*, pages 9572–9582. IEEE, 2019.
- [13] Mehrdad Hosseinzadeh, Ramin Fahimi, Yang Wang, et al. Unsupervised learning of camera pose with compositional re-estimation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 11–20, 2020.
- [14] Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. Tartanair: A dataset to push the limits of visual slam. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4909–4916. IEEE, 2020.
- [15] Leslie N Smith and Nicholay Topin. Superconvergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, pages 369–386. SPIE, 2019.
- [16] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.
- [17] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems*, 30, 2017.
- [18] Zheng Huai and Guoquan Huang. Robocentric visual-inertial odometry. *The International Journal of Robotics Research*, 41(7):667–689, 2022.
- [19] Joan Sola. Quaternion kinematics for the error-state kalman filter. *arXiv preprint arXiv:1711.02508*, 2017.
- [20] Zichao Zhang and Davide Scaramuzza. A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7244–7251. IEEE, 2018.
- [21] Patrick Geneva, Kevin Eickenhoff, Woosik Lee, Yulin Yang, and Guoquan Huang. Openvins: A research platform for visual-inertial estimation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4666–4672. IEEE, 2020.
- [22] Jeffrey Delmerico and Davide Scaramuzza. A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 2502–2509. IEEE, 2018.

Frustumbug: a 3D mapless stereo-vision-based bug algorithm for Micro Air Vehicles

R.S. Meester^{*}, T. van Dijk^{*}, C. De Wagter[†], G.C.H.E. de Croon^{‡†}

Control and Simulation, Faculty of Aerospace Engineering
Delft University of Technology, The Netherlands

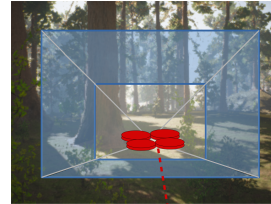
ABSTRACT

Obstacle avoidance is an important capability for flying robots. But for robots with limited resources, such as small drones this becomes particularly challenging. Bug algorithms have been proposed to solve path planning with only minimal resources. And stereo vision provides a rich description of the world for limited weight but typically has a limited Field of View (FoV) and is fixed to the drone frame to further reduce weight. Based on these, a computationally light 3D path-planning algorithm is proposed. The proposed algorithm is called Frustumbug and is based on the Wedgebug algorithm since this algorithm copes well with a limited FoV. Since Wedgebug only addresses 2D problems, the Local- ϵ -Tangent-Graph (LETG) is used to extend the path planning to 3D. Disparity images are obtained through an optimized stereo block matching algorithm. Frustumbug copes well with noisy range sensor data and includes 3D trajectories like reversing, climbing and descending maneuvers to avoid or escape local minima. The algorithm has been tested with 225 flights in two challenging simulated environments and achieved a success rate of 96%. Here, 3.6% did not reach the goal and 0.4% collided. Frustumbug has been implemented on a 20-gram stereo vision system and was tested in the real world on a MAV. This shows the potential for small drones to reach their targets fully autonomously based on very limited resources.

1 INTRODUCTION

The drone market keeps growing, with applications ranging from industrial inspection to search-and-rescue and package delivery. Since most of the applications have to be performed autonomously, autonomous obstacle avoidance is of the essence. Current solutions combine multiple sensors and are computationally expensive and memory intensive. An efficient, lightweight solution would require less energy from the battery, increasing the flight time while reducing weight requirements.

Over the past few decades, several distance measurement sensors have been studied. To allow for onboard path planning, vision-based sensors are the only promising candidate when looking at size and weight and power [Aswini et al.,



(a) Drone's viewing frustum



(b) Depth image obtained from the information inside frustum

Figure 1: Creation of the 3D viewing frustum. The drone is represented in red. The red dotted line indicates the traveled path. Depth image includes a safety margin around obstacles.

2018]. To limit the computational resources used by the drone, a mapless method is preferred over a map-building method. Since the outdoor flight is assumed, building maps of large, complex environments requires high resources.

Ideally, the generated path should resemble the globally shortest path. Therefore, the drone's ability to move up or down to avoid obstacles needs to be exploited, hence this paper focuses on path planning in three dimensions. 3D mapless vision-based methods exist but are often combined with probabilistic roadmaps [e.g. Matthies et al., 2014 and Lee et al., 2021] or machine learning algorithms [e.g. Doukhi and Lee, 2022 and Grando et al., 2022]. Since there is a certain amount of randomness in these methods, the choice of paths can not always be explained, and may not always lead to the goal. Furthermore, these methods tend not to include strategies for escaping local minima, or use randomly generated sub-optimal paths to escape [e.g. Matthies et al., 2014 and Yu et al., 2018].

Bug algorithms are used for robots with limited computational resources since they plan their path using very little computation and memory. Another advantage is that these finite-state machines provide a strategy for escaping local minima. Wedgebug [Laubach and Burdick, 1999] is a two-dimensional bug algorithm that solved the problems for robots with a small Field of View (FoV), by rotating the sensor when more information is required. However, to create an accessible algorithm for the many drones and cameras without that capability, and to limit the necessary moving components, the algorithm should be designed without the requirement of camera rotation. When stuck in a local minimum, Wedgebug will follow the obstacle boundary until an intermediate waypoint is found which decreases the distance to the goal [Laubach and Burdick, 1999]. However, in 2D an

^{*}J.C.vanDijk-1@tudelft.nl

[†]c.dewagter@tudelft.nl

[‡]g.c.h.e.decroon@tudelft.nl

obstacle can only be avoided in two directions: left or right. In 3D, the number of directions to avoid an obstacle increases to infinity, since any location on the obstacle boundary can be chosen. 3DBug [Kamon et al., 1996] reduces the necessary calculations by discretizing the continuous obstacle contour into a set of points. However, 3DBug assumes an infinite sensor range and a 360 degrees FoV. Furthermore, it has been tested on relatively simple environments where the locally best decisions mostly lead to the globally best paths.

The goal of this study is to create a 3D path planning algorithm, which can navigate a drone to a target while minimizing deviation from the nominal path using limited resources. We contribute to the existing literature by; (i) extending Wedgebug to operate in 3D, (ii) introducing additional navigation states for avoiding and escaping local minima, (iii) designing a novel waypoint selection method to increase robustness to noisy range sensor data, (iv) extensively testing the proposed algorithm in simulation and real-world, using a very light-weight stereo vision system. We will call it Frustumbug, as the frustum can be seen as the 3D extension of a wedge. Figure 1 shows how the viewing frustum is shaped (Figure 1a), along with its corresponding configuration space (Figure 1b). Our contributions take the next step towards a publicly available self-contained path planning package and allow for future extensions since each path planning decision follows from a logical decision captured in a decision tree, and each branch can be altered individually. Assumptions include outdoor flight during daylight, where GPS signal is available. Furthermore, the target and obstacles are assumed static and maze-like environments are not considered. The remaining part of the paper is structured as follows: first, section 2 shows the related work and section 3 explains how the chosen approach is implemented. Then, section 4 presents the experimental results from both simulation and real-world testing. This paper then discusses the results in section 5 and concludes in section 6.

2 RELEVANT WORK

The relevant work is divided into two sections: we first look at the literature on distance measurement sensors to optimize our sensing, followed by literature on avoidance algorithms to complete the path planning package.

2.1 Sensing

Vision-based sensors are suitable for outdoor environments since they usually have enough texture to be captured with a camera [Matthies et al., 2014]. Deep/Machine learning algorithms are not considered for obtaining depth information, since these methods consist of multiple convolutional layers, which require large matrices to be stored and many computations to be made [e.g. Doukhi and Lee, 2022]. Moreover, the parameters of a neural network result from training the algorithm for a longer period of time. If it shows inconsistent results for repetitive image texture or low-textured regions, the parameters can not be easily adapted to improve

the result. It would have to be trained again on an improved dataset, or with different hyperparameters.

Vision-based sensing can be done with one, two or more cameras. Monocular vision in obstacle avoidance is generally paired with optical flow methods. These have their drawbacks in obtaining disparity images compared to stereo vision: a larger image area needs to be searched to find matching pixels, estimates on position and attitude changes need to be included in the calculation and it is not able to sense distances in the direction of travel, since there is no optical flow in the focus of expansion [van Dijk, 2020]. Following the argument above, stereo vision is preferred since the added weight and required power are small; they can be as lightweight as 4 grams and be run on 168 MHz microprocessors [McGuire et al., 2017]. To limit the computational resources, a solution using more than two cameras is not considered.

Stereo matching can be done either local or global, and sparsely or densely. Although global methods tend to generate better results in low-texture environments, it comes at a computational cost [Liu et al., 2020]. Furthermore, outdoor environments during daylight have enough texture for local methods. Sparse methods only calculate the disparity for interesting features, which decreases the computational cost. However, this would leave large gaps in the disparity image, which are unwanted. Therefore, a local dense stereo matching algorithm will be used to create the disparity image. Possible stereo-matching algorithm candidates can be compared by their run-time, used platforms, performance and code availability from common benchmarks KITTI [Geiger et al., 2012] and Middlebury [Scharstein and Szeliski, 2002]. The most promising candidates are Block Matching (BM) [Scharstein and Szeliski, 2002] and Semi-Global Block Matching (SGBM) [Hirschmuller, 2005], due to their fast execution time and performance in outdoor environments [Lyrakis, 2019]. In addition, it was concluded that BM is preferred over SGBM, since it is faster and does not have the tendency to fill the sky with incorrect large disparity values [Lyrakis, 2019].

Block Matching can be performed using various metrics for similarity, pre-/post-filters and block sizes. The Sum of Absolute Differences (SAD) consists of relatively simple calculations, resulting in a fast algorithm [Patil et al., 2013]. Filters are used to delete outliers, which can be caused by occlusions or untextured or repetitive image regions. These filters will be discussed in more detail in subsection 3.1.

2.2 Avoidance

After the obstacles have been observed by the stereo camera, a path needs to be planned around them for avoidance. Path planning can be done by using either map-based or mapless methods. Map-based methods either start with a complete map of the environment (global map), or build them using sensors (local map), here referred to as map-building [Elmokadem and Savkin, 2021]. Mapless methods do not

utilize this previously gathered information and show reflex-like behavior based on current sensor data [Elmokadem and Savkin, 2021]. Since complete maps of outdoor environments are hardly ever available, map-based approaches which require a global map are unrealistic for outdoor path planning.

There is a trade-off in the decision for a map-building or mapless method. The objective of this research is to minimize deviation from the nominal path using limited resources. Map-building methods require more computational resources and memory, whereas mapless methods make reactive decisions using a relatively small amount of processing of the sensor data [Elmokadem and Savkin, 2021]. However, when stuck in a local minima, a mapless method does not use any information of previously sensed obstacles. It needs to scan the obstacles again and it could fail to identify an escape path already known to the map-building method. Hence, map-building methods could lead to more optimal paths. Nevertheless, only a small amount of dead ends is expected in outdoor environments, since obstacles can also be avoided by passing over them. Therefore, a mapless approach is chosen to limit computational resources.

Bug algorithms are computationally cheap mapless path planning algorithms, due to their simple reasoning in finding a way to the goal. For example, Bug2 [Lumelsky and Stepanov, 1986] draws a main line, or *M-line*, from start- to goal position and follows this line until an obstacle is encountered. The obstacle boundary is then followed until the *M-line* is met, and the motion to the goal is resumed. Wedgebug [Laubach and Burdick, 1999] is of interest because it uses a range sensor with a small FoV, and we also have limited FoV vision. It scans a 'wedge' to see if the goal is safe, or if an intermediate waypoint needs to be set. In case there is no solution inside the wedge, it will scan an adjacent wedge to obtain more information. This algorithm was used to move a planetary rover horizontally, i.e. in 2D. Extending it to 3D is necessary to utilize the drone's ability to move vertically. 3DBug [Kamon et al., 1996] is a bug algorithm that extends path planning to 3D by constructing a Local- ϵ -Tangent-Graph (LETG) which discretises the obstacle boundary. This is necessary since any obstacle boundary is a line with an infinite number of points. 3DBug assigns a finite number of points to the boundary and creates the LETG, which consists of the lines connecting the current- and the goal position to these waypoints, as shown in Figure 2. Figure 2a shows an example for a 2D plane and Figure 2b for a concave obstacle.

A reactive approach requires each disparity image to be used individually for path planning. Since the assumption that the robot can be modeled as a point robot does not hold, a safety margin needs to be included. This can be done with low computational costs and little memory by expanding the obstacles in disparity space to construct the configuration space (C-space) [Matthies et al., 2014], as presented in Figure 3. Each pixel in the disparity image is translated into its world coordinates and a sphere of a predefined expansion

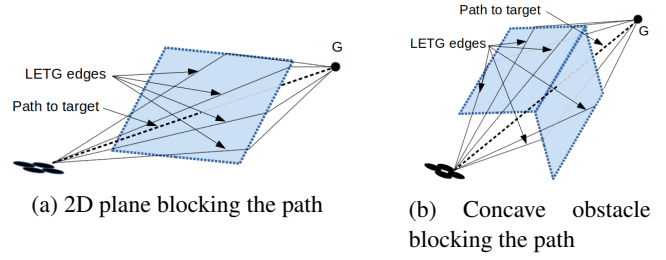


Figure 2: LETG edges used by 3DBug for an obstacle blocking the drone's path to goal, represented by the thicker line.

radius is drawn around it. An example for one pixel is shown in Figure 3a. Next, a rectangle that hides the sphere behind it is drawn just in front of the sphere, to simplify calculations. The area captured by each rectangle obtains a depth value equal to the pixel's original depth value minus the expansion radius. Using the C-space, the drone can be modeled as a point and it can safely choose an obstacle-free pixel in the disparity image, as shown in Figure 3b.

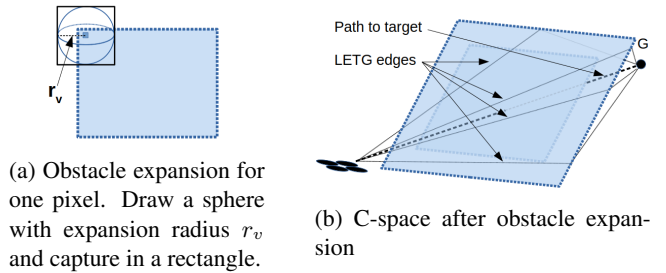


Figure 3: Obstacle expansion in disparity space using expansion radius r_v , creating the drone's C-space.

3DBug makes two assumptions that do not hold: The range sensor has perfect readings and all obstacles can be modeled as polyhedral obstacles. The stereo camera will not provide perfect readings, which makes it difficult to model the obstacles as polyhedral. As a solution, a boundary tracing algorithm can be used to find the obstacle contours in the disparity image. The Moore-Neighbor tracing algorithm is often used, since it is fast and easy to implement [Reddy et al., 2012]. After creating the contour, the point that represents the locally shortest path can be calculated. Since boundary tracing could identify the (unsafe) concave edge in Figure 2b as safe, sudden points are searched within the image. These are defined in Pointbug [Buniyamin et al., 2011] as large sudden changes in depth readings. Hence, the sudden points will ensure that the drone chooses a waypoint around the obstacle instead of into it.

If the drone is stuck in a local minimum, it needs to apply a strategy to escape. Wedgebug halts the robot and scans additional wedges to find an intermediate waypoint [Laubach and Burdick, 1999] and goes into boundary-following mode. Here, the robot skirts the contour of the obstacle and leaves the boundary whenever the goal direction is free of obsta-

cles. Other escaping strategies are proposed by potential field methods, for example introducing virtual obstacles that push the drone away [Lee and Park, 2003] or using rotational forces to steer around the obstacle [Sfeir et al., 2011]. However, the literature focuses on solutions for 2D problems, and 3D mapless obstacle avoidance algorithms tend not to mention they are getting stuck in local minima [e.g. Lee et al., 2021 and Oleynikova et al., 2015]. Due to the limited FoV of stereo vision also in the vertical direction, situations in which the drone cannot instantly see a vertical escape route can definitely occur.

3 METHOD

The method section first discusses the sensing, followed by the intermediate waypoint selection method and the finite-state machines. The latter is divided into the description of the Wedgebug states, changes in these states and the additional states for avoiding and escaping local minima.

3.1 Sensing

The rectified stereo image pair is fed to the stereo matching algorithm to generate the disparity image. The depth in meters can be obtained by multiplying the stereo base by the focal length, and dividing by the disparity in pixels. Figure 4 shows the result of the stereo BM algorithm using SAD. The RGB image in Figure 4a is taken from the simulated environment UrbanCity¹. Its corresponding true depth image is shown in Figure 4b. The generated depth image is shown in Figure 4c shows the result using BM's default parameters. To improve the result, a grid search is performed to find the parameter combination giving the best disparity images. For this, a dataset of 100 images with ground truth depth values is generated from the open-source simulation software AirSim². To quantitatively compare the performance of the parameter combination, each disparity image was evaluated for completeness, accuracy and noise. The parameters are for block size, texture filter, uniqueness ratio filter, speckle filter, left-right consistency check. An extra filter has been added in an attempt to further reduce noise: morphological opening. The generated depth image using these optimized parameters is shown in Figure 4d.

The next step is to expand the obstacles in disparity space, to create the C-space. As suggested by the authors, look-up tables for obstacle expansion coefficients are generated pre-flight for efficiency [Matthies et al., 2014]. The method is shown in Figure 5. Figure 5a is a small cutout of the optimized parameters' depth image showing two pixels as an example of obstacle expansion. The width of r_v has to be at least half the width of the drone to include a large enough safety margin. Expanding every pixel of an image with a pixel resolution of 240x320, results in the C-space shown in Figure 5b. To further improve performance, pixels are only

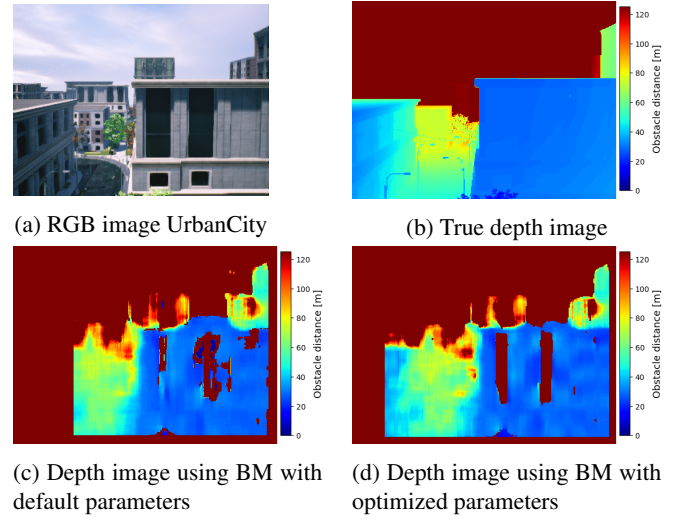


Figure 4: True- and stereo Block Matching depth images, belonging to an RGB image from simulated environment UrbanCity.

expanded if their depth value is below a threshold, and the images are downsampled. In the example of Figure 5c, a depth threshold of 30 meters is used, speeding up the runtime by a factor 2.5. Figure 5d shows the image downsampled by a factor 100, yielding a 24x32 resolution, and is 110 times faster than the complete C-space. The thresholded and downsampled image shown in Figure 5e is 210 times faster than the complete C-space, with satisfactory results. Figure 5f shows what happens to the C-space if the BM parameters are not optimized: the noisy pixels on the nearby building cover the entire FoV after obstacle expansion, making path planning impossible.

The best parameters for a dataset are not necessarily the best parameters for each individual image. For example, some images could have highly repetitive textures and would need different values for the uniqueness ratio parameter. Therefore, if the drone is not able to move due to poor stereo matching, Frustumbug will change the uniqueness ratio temporarily, in an effort to escape the current location.

3.2 Waypoint selection

Wedgebug and 3DBug are both finite-state machines (FSM). They set intermediate waypoints to avoid obstacles, and move to the goal position when possible. The basis of the avoidance strategy is shaped by Wedgebug.

The novel waypoint selection method, shown in Figure 6, finds the locally best waypoint based on its sensor data. The goal- or current waypoint pixel must always be located inside the FoV, else the drone will hover and scan. The goal pixel is shown in white in the middle of each image. In this example, the goal is blocked by an obstacle at around 30 meters and a waypoint with a clearance of at least 35 meters will be selected. The first step is to identify the 'safe' pixels, as shown by the white mask in Figure 6a. Second, the mathe-

¹<https://www.unrealengine.com/marketplace/en-US/product/urban-city>

²<https://microsoft.github.io/AirSim/>

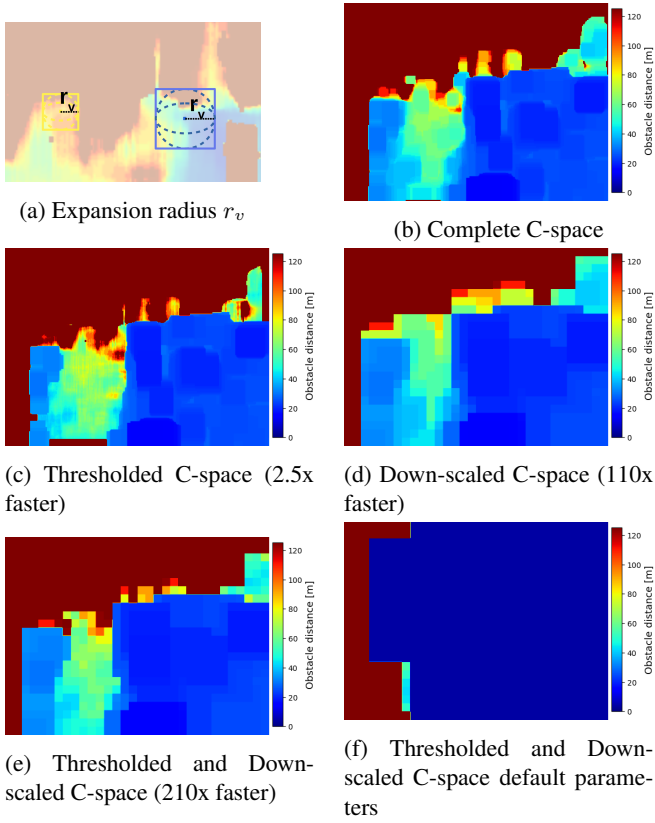


Figure 5: Comparing C-space efficiency measures results

mathematical erosion of this mask is performed, yielding the eroded mask in Figure 6b. Since the obstacle expansion in disparity space is extremely sensitive to noise, as shown in Figure 5f, its expansion radius can not be too large. The mathematical erosion is performed to ensure a large enough safety margin, in a noise-robust manner. Third, the edge of the eroded mask is calculated, shown in Figure 6c. To get the locally shortest path, the obstacle needs to be avoided with the smallest deviation possible. Hence, the best waypoint will be located on the edge.

Since the Moore-Neighbor Tracing Algorithm requires a starting point for each boundary it traces, the novel method is preferred because it is able to find multiple safe areas. In addition, it makes it simpler to include an extra safety margin. Since a mask also makes no distinction between waypoints around or into concave obstacles, the fourth step is to look for sudden points. These are defined as pixels that have neighboring pixels have a change larger than 20% in their depth value, and are shown in Figure 6d.

The last step is to find the best waypoint according to a cost function. For this, the obstacle boundary is discretized to pixel resolution: each pixel indicates a new possible waypoint. The cost function minimizes the pixel distance to the sudden points within the eroded mask (if they exist) and the goal pixel, shown in Figure 6e. Wedgebug places the waypoint at a finite sensor range [Laubach and Burdick, 1999].

Frustumbug places the waypoint beside the obstacle it is avoiding, to allow for goal scanning upon arrival.

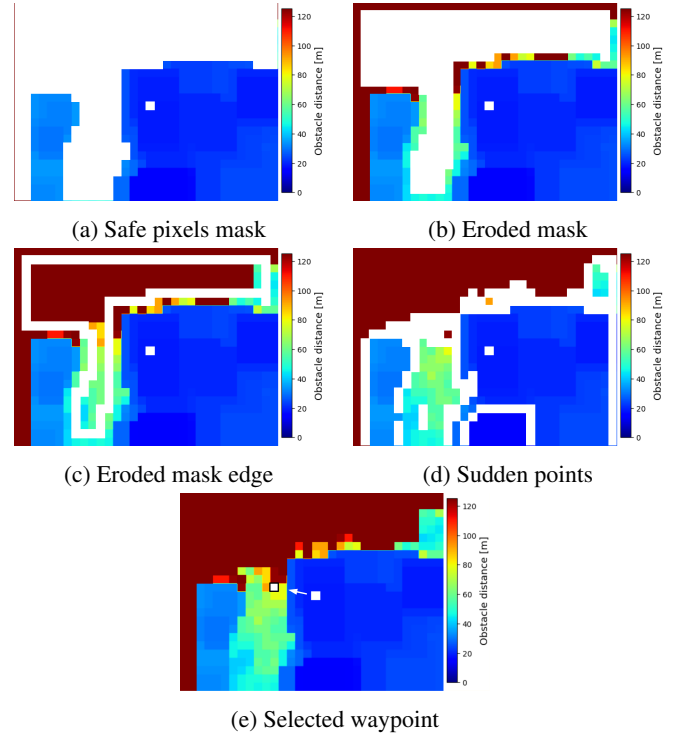


Figure 6: Waypoint selection procedure

3.3 Finite-state machine - Wedgebug states

Similar to Wedgebug and 3DBug, Frustumbug is also a finite-state machine. A simplified version is shown in Figure 7. The Wedgebug FSM is shown by the blue, green and yellow shapes, which are connected by solid lines. Frustumbug expands the FSM by adding reversing, climbing and descending states, indicated by the dashed grey lines. These can be activated from motion to goal, motion to waypoint or boundary following. The states with a solid boundary indicate a moving robot, whereas the dotted boundaries indicate a stationary robot, used for scanning. For the drone, stationary means hovering.

Wedgebug uses state *Motion to Goal* (*MtG*) to move the robot directly towards the goal. If the goal is reached, the algorithm halts. If an obstacle is detected, a new waypoint will be searched using the method described above. If found, state *MtG* transitions to state *Motion to Waypoint* (*MtW*). If not, it transitions to state *Scanning Waypoint* (*SW*). State *MtW* moves the robot to the waypoint and checks if state *MtG* is possible when the waypoint is reached. If the path to the waypoint becomes unsafe because of a previously unseen obstacle, a new waypoint is searched. If this waypoint is not found, state *MtW* transitions to state *SW*. In other words, state *SW* is activated when there are no safe waypoints visible in the current wedge. The robot halts and scans additional

wedges to find a new waypoint that can still decrease its distance to the goal. If no such waypoint can be found, state *SW* transitions to state ***Scanning Boundary (SB)***. State *SB* keeps scanning additional wedges until a new waypoint is found. If none are found, the goal is deemed unreachable. Otherwise, state *SB* transitions to state ***Boundary Following (BF)***, which moves the robot around the obstacle boundary until a transition to state *MrG* is possible, or until there exists a leaving point: a waypoint which has a smaller distance to the goal than the previously visited coordinates, making a transition to state *MrW* possible [Laubach and Burdick, 1999]. The algorithm stores the direction in which the obstacle boundary is being followed (CW/CCW) to prevent backtracking. If the goal is reachable, Wedgebug guarantees global convergence [Laubach and Burdick, 1999].

3.4 Finite-state machine - Changes to Wedgebug states

Since Frustumbug obtains depth information from imperfect sensor readings, it is not able to identify independent obstacle boundaries. In other words, it is not sure whether it has been following the boundary of the same obstacle, or if it has jumped to different ones (e.g. it is hard to identify which branch belongs to which tree in a dense forest). This causes Frustumbug to lose its global convergence guarantee. Moreover, if an unseen obstacle appears within a short time after transitioning from state *BF* into state *MtW*, it cannot be said with certainty if this is the same obstacle whose boundary it was just following, or if it is a new obstacle. If it is the same, it should follow the boundary in the same direction (CW/CCW). Therefore, the positive direction around an obstacle is remembered until a distance is traveled that exceeds a threshold, currently implemented as 20 meters.

Frustumbug is designed for a fixed stereo camera, hence it can not rotate its sensors like Wedgebug. Whenever it has to scan an extra wedge, it will rotate the drone around the vertical axis (yaw). To minimize the chance of crashing into obstacles while the camera is facing away from the direction of motion, this is only performed when hovering stationary. If the drone is moving, it has to come to a stop first. To transition from state MtW to state MtG , the drone first needs to stop and scan the goal. This is done via the state **Scanning Goal** (SG). Furthermore, it required a design change for state BF , since it cannot scan the goal direction while moving along the boundary. Frustumbug divides state BF into two states: **Boundary Following - Waypoint** (BFW) and **Boundary Following - Turning** (BFT). When state SB has found a waypoint, it transitions to state BFW , which moves the drone to the waypoint. Upon arrival, there is a transition to state BFT , which rotates the drone to scan the goal and/or the obstacle boundary to plan a new waypoint. The distance between the waypoints results from a trade-off between stopping too often and missing opportunities where the goal direction was safe. Unfortunately, states BFW and BFT can lead to long paths if they miss safe goal direction opportunities and the continuous

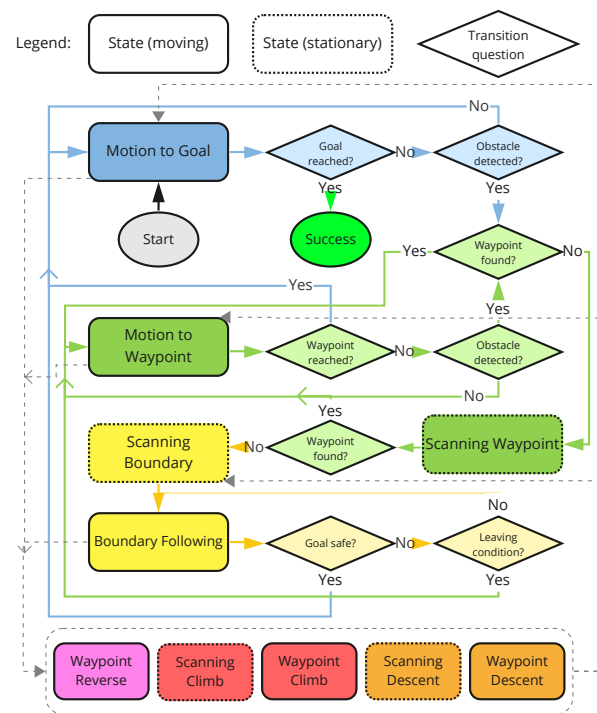


Figure 7: Simplified Frustumbug FSM

stopping, rotating and going can be time-demanding.

To decrease the frequency in which states *BFW* and *BFT* are activated, state *SW* is allowed to scan for waypoints that do not necessarily decrease the distance to goal, as long as they are within a 180 degrees FoV when facing the goal. Since Frustumbug already lost its global convergence guarantee due to being unable to identify independent obstacle boundaries, this rule is implemented to improve the overall performance. Further improvements are made by allowing state *SW* to always scan a left wedge and a right wedge, unless a fixed direction around the obstacle has been specified. This is implemented to prevent missing out on good opportunities to avoid the obstacle. Wedgebug would stop scanning as soon as it finds a waypoint in the left or right wedge [Laubach and Burdick, 1999].

To prevent looping around the same obstacle, Wedgebug has implemented a loop detection: when a location on the obstacle boundary is visited twice, a loop is detected [Laubach and Burdick, 1999]. A two-dimensional loop might not be found in a three-dimensional environment, since the shape of an object is not always constant over height, and the altitude of the drone can change during state *BFW*. Since Frustumbug can not clearly distinguish independent obstacle boundaries, it is possible that it will follow the boundary of multiple obstacles. Hence, even if a loop is found, the maneuver can be time-consuming. Therefore, the state *MtG* is activated by state *BFT* once the *M-line* is crossed. Instead of using the *M-line* from start- to goal position, Frustumbug defines this

line as the line between the start of the state *SB* and the goal position, since the *M*-line does not play a role in the other states.

3.5 Finite-state machine - Additional states

To further decrease the frequency in which states *BFW* and *BFT* are activated, Frustumbug introduces the possibility to reverse the drone backward along previously visited GPS points using state **Waypoint Reverse** (*WR*). This state looks for escape points that would steer the drone around the obstacle in an earlier stage, using different parameter settings. Perhaps it did not see the obstacle before due to a stereo mismatch. Just in case it is mismatched again, the obstacle location is remembered as a single pixel, which is redrawn in the incoming images while reversing. One pixel is enough since it will be expanded in disparity space to create the C-space. State *WR* is only activated if the previously visited GPS coordinates are located on a straight line, since the drone is going there blindly and turning radii can be different than on the way there.

Next, a distinction has been made regarding obstacle size when reaching local minima: the C-space could be blocked by a farther away large obstacle (note that these pixels are still unsafe) or a nearby smaller obstacle which caused the obstacle expansion to cover the entire FoV, similar to what happened in Figure 5f. In case of a small obstacle, it is preferred to reverse and avoid using state *WR*, or to find a new waypoint using state *SW*. However, for a large obstacle, a new waypoint may not be found by states *WR* or *SW*. Instead, the altitude is increased in an effort to avoid the obstacle by going over it.

The process for selecting the best waypoint for climbing is shown in Figure 8. Due to the limited FoV it is not possible to move straight up, hence the drone will climb using a small flight path angle. To limit the required memory and computation when scanning its surroundings, only three pixel rows are checked, shown in white in Figure 8a. In the 32x24 image, these are rows 3, 6 and 9. The waypoint will not be selected in the top rows (0, 1 and 2), since it would move out of FoV quickly when the drone pitches forward to move, and obstacle presence could not be checked. For each white row, the best pixel's column is stored if the number of safe pixels exceeds the threshold for a safe climb. The best pixel's column is the column with the largest distance from any obstacle in that row. Figure 8b shows the safe pixels in grey, as well as the best pixel in white. If a safe pixel exists in the highest row, it is chosen first, followed by the middle row and then the bottom row. Preference is given to the climbing directions that are closer to the goal direction.

The goal direction is always scanned first for a possible waypoint to climb. If not found, state **Scanning Climb** (*SC*) is activated. It will first scan the 180 degrees in the goal direction to find a suitable climbing waypoint. If no waypoints are found, it will scan the remaining 180 degrees. If a way-

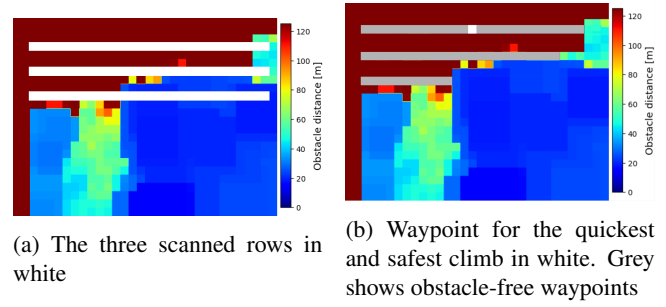


Figure 8: Method for selecting the best waypoint for a climbing maneuver

point is found, state **Waypoint Climb** (*WC*) will move the drone towards this waypoint. Else, state *SB* is activated. A climbing maneuver always consists of at least 2 segments to minimize the deviation from the nominal path: halfway the desired height difference, a new waypoint will be searched. It is possible that Frustumbug uses more than 2 segments to obtain the desired height difference if a previously unseen obstacle blocks the waypoint in state *WC*. A new segment is then added, using the method described above.

The climbing maneuver is created to allow the drone to avoid obstacles by flying over them, followed by a descending maneuver. Therefore, the drone is instructed to keep flying at a higher altitude until the obstacle is passed. Since individual obstacle boundaries can not be distinguished, this is implemented as a minimum distance to be flown toward the goal at a higher altitude. Once arrived, the goal direction is always scanned first for a possible waypoint to descend. If not found, state **Scanning Descent Forwards** (*SDF*) is activated. The method is similar to selecting climbing waypoints, but rows 14, 17 and 20 are scanned for descent.

State *SDF* only scans the 180 degrees FoV towards the goal. This state is normally used after the obstacle has been passed because descending backward could send the drone back to the wrong side of the obstacle. If no waypoint for descent is found, the drone will keep flying towards the goal at the same altitude for a few meters and transition to state *SDF* again. Else, state **Waypoint Descent** (*WD*) will move the drone towards this waypoint. Upon arrival, the goal direction will be scanned to see if the goal is inside the FoV. If not, state **Scanning Descent Backwards** (*SDB*) will be activated, which will scan the 180 degrees in the opposite direction of the goal. If a waypoint exists here, state *WD* is activated to execute the path. Else, state *SDF* is activated, which will try to find a waypoint to descend the drone toward the goal.

In case the drone is located nearby and above the goal, or if state *BFT* noticed that the goal is located below its FoV, it has no clear preference between states *SDF* or *SDB*. Hence, state **Scanning Descent Either** (*SDE*) is designed such that it will first scan in the goal direction. If no waypoint is found, it will continue to scan until a waypoint is found or until a full revolution is completed, similar to what state *SC* does

for climbing. A note to this: if the drone realizes it is above an obstacle (e.g. a roof), it will keep moving towards the goal until this obstacle is passed. It attempts to identify this situation by checking if the pixels blocking the goal have a roughly linearly increasing depth value since this is usually the case when above buildings or large trees.

In total, the drone can adapt to 14 different states on its way to the goal position. We are aware that these FSMs can come across as quite elaborate. However, in complex environments, there are many possible failure modes that need to be accounted for, and we would like to present the states in detail to keep a clear overview of the drone's behavior during flight. Note that each path planning decision can be changed easily, allowing for quick user-preferred changes or upgrades.

4 EXPERIMENTAL RESULTS

The proposed algorithm has been tested both in simulation and on a real drone. First, results from simulation environments UrbanCity and Forest are presented, followed by the real-world results. A number of start- and goal positions are generated for both UrbanCity and Forest environments. The generated paths are stored, from which the success rate and the path length can be obtained.

4.1 Simulation UrbanCity

UrbanCity is an environment built in Unreal Engine 4. Open source project UnrealCV included several commands to interact with the environment. The environment simulates an urban city corner and contains buildings, trees, roads and street signs. Start- and goal positions are chosen throughout the environment: 11 on ground level and 3 on top of buildings. These 14 locations yield 182 possible paths to be traveled. Since UnrealCV is only able to set static camera poses, drone dynamics are not included. To navigate through the environment, the camera location is moved by 1 meter in the direction of the waypoint each time. Only the yaw angle is changed, not pitch nor roll.

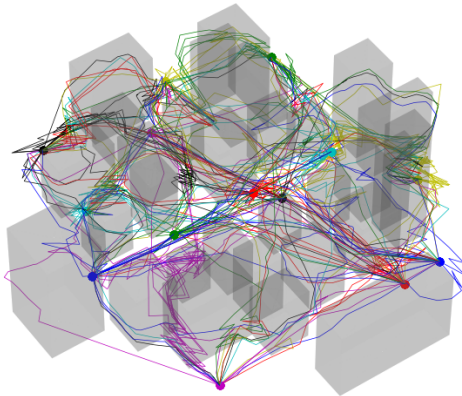


Figure 9: All successful paths in UrbanCity. The paths are color-coded similar to their goal position.

In total, 177 out of 182 goals were reached successfully,

leading to a success rate of 97.3 %. Figure 9 shows all the successful paths. Each path has the same color as its goal, however since there are only 7 colors for 14 goals, every color appears twice. The 5 unsuccessful paths are shown in Figure 10, of which the first two are shown in Figure 10a. Here, the crosses represent the starting points, the dots are where it failed and the stars are the goals. The black line used state *WC* to avoid an obstacle at a higher altitude near the end. Upon arrival, the goal was still inside FoV near the bottom, but was blocked by the top of the building, hence state *MtW* was activated. A small distance later, the goal went out of FoV and state *SDE* was activated, which found a waypoint in the direction it just came from: in the front of the building. The drone got stuck in this loop until a timeout was reached. The green line collided with a building, due to a temporary change in value for the uniqueness ratio parameter, as explained in the last paragraph of subsection 3.1. After a turn, the drone was facing a different obstacle, which could not be detected using the temporarily increased uniqueness ratio. However, without this change in the uniqueness ratio parameter, dozens of simulations got stuck.

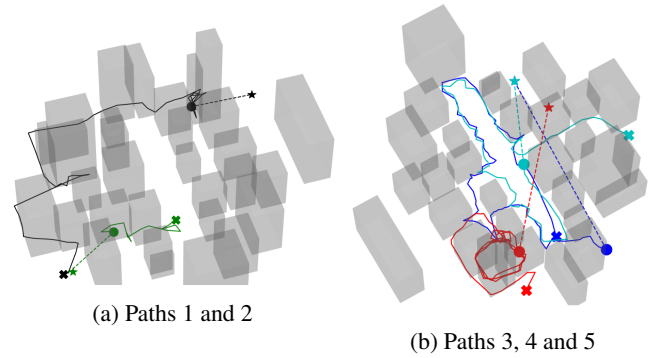


Figure 10: All unsuccessful paths in UrbanCity. A cross represents a starting point, a star represents the goal and the dot is where the simulation failed.

The three other unsuccessful flights, shown in Figure 10b, all ended in state *BFT* or *BFW* at the timeout. The blue and cyan lines could not find a path in between the buildings and continued to follow the boundary of a growing group of obstacles. The red line initially starts boundary following almost from the start, but gets stuck in a loop around one of the higher buildings. All three did not cross their respective *M*-lines. They initially did not start climbing because they were looking into small alleys, meaning that the minimum percentage of nearby pixels was not reached.

In subsection 3.5 a distinction between small and large obstacles was made, which would both have the entire FoV covered after obstacle expansion. Figure 11 shows an example to explain both possibilities using on-board data. Figure 11a shows a nearby light pole and Figure 11b shows a farther away building. This is supported by their depth maps:

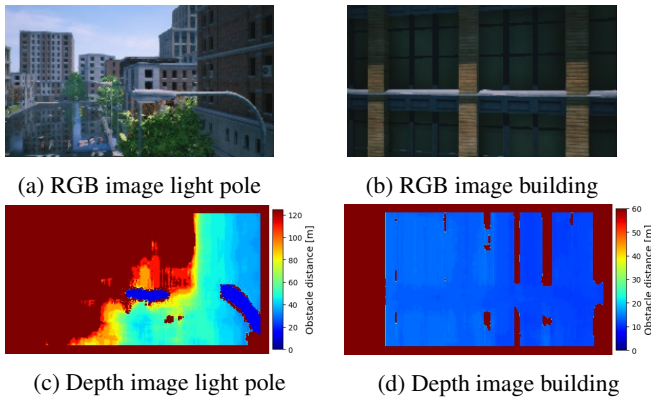


Figure 11: Examples of nearby (light pole) and farther away (building) obstacles, both blocking the C-space. The light pole would trigger a transition to either state *SW* or *WR*, depending on the current state. The building would trigger a transition to state *SC*.

Figure 11c shows only a few nearby pixels in blue, whereas Figure 11d shows a large obstacle. After obstacle expansion, the nearby pixels cover the complete FoV for both. To transition to state *SC*, the percentage of nearby pixels in the depth image needs to exceed a threshold (currently 80 percent), else there will be a transition to either *SW* or *WR*, depending on the current state. For Figure 11a, state *SW* was activated, because the camera had just turned and there was no straight line segment, so the transition to state *WR* was not allowed. For Figure 11b, state *MTG* transitioned into state *SC*.

The path lengths are compared to their default path length. Note that the default path is not equal to the globally optimal path. Since it is complicated to get the globally shortest path out of UrbanCity, the default path length is defined by the path length that is obtained if the drone would simply climb high enough, fly over all obstacles, and descend to the goal, using the FoV restricted climbing maneuver. On average the path lengths are 75.8 % of their default path lengths, meaning that Frustumbug has found an average shortcut of 24.2 %. Results are shown in Figure 12. All 6 flights exceeding 115 % have activated states BFW and BFT during flight.

4.2 Simulation Forest

The open-source simulator AirSim is developed by Microsoft and contains an outdoor environment called Forest. It includes many trees, branches, bushes, rocks and hills, and is shown in Figure 13. The green dot indicates the starting position, and the 8 orange dots indicate the goal positions. Some of the goals are located near trees, but all are reachable. The paths are flown at 3 different altitudes, and from low to high altitude and vice versa, resulting in 40 paths. The area with the purple dots consists of fewer trees and more hills, hence it is used to test Frustumbug's ability to deal with ground elevation changes. In total, 43 paths are generated. The arrows indicate that the waypoint location is

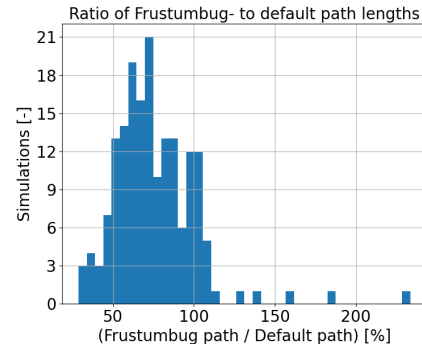


Figure 12: Frustumbug's path lengths histogram

outside of the image. Drone dynamics are included in this simulation, making it more representative for outdoor flight.



Figure 13: Forest environment. The green dot represents the start position, the orange and purple dots show all the goal positions. Arrows indicate they are off the map.

39 out of 43 goals were reached successfully, leading to a success rate of 90.7 %. Figure 14 shows all the successful paths and Figure 15 shows the 4 unsuccessful paths. Figure 14a starts with the top view. Note that the left side of this plot is the most cluttered part of the environment, with a large number of trees located close together. Figure 14b shows the 24 paths where start- and goal positions have the same altitude. Two goals (14 and 15) are not reached, both in the cluttered area. Figure 14c shows the 16 paths that go from low to high altitude and vice versa. Here, two goals (34 and 38) were not reached, of which one was located in the cluttered area. Figure 14d shows the three waypoints far into the forest, in the area containing hills. All three were reached by the proposed algorithm.

Ideally, the generated path lengths are compared to the globally shortest path lengths. However, calculating these would require accurate obstacle data, which is not available. Furthermore, the climbing and descending method presented in UrbanCity would yield unrealistically long default paths, since the trees are generally not avoided by climbing over them. Therefore, the paths had to be analyzed qualitatively. It was found that any deviation it took from the nominal path

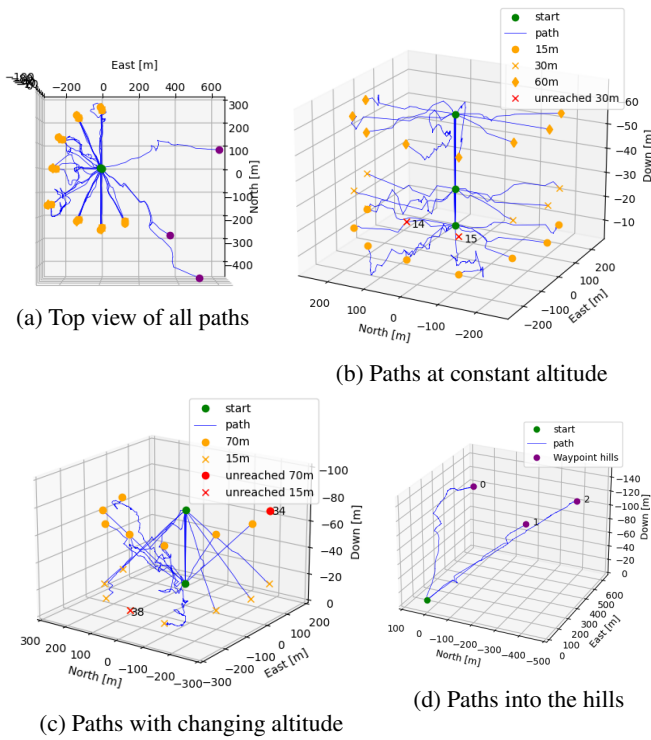


Figure 14: All successful flights in Forest

was supported by obstacle presence.

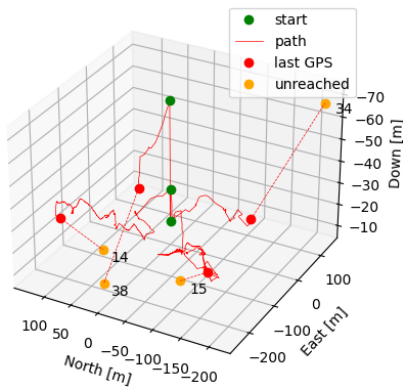


Figure 15: Four unsuccessful paths in Forest

The four unsuccessful paths are caused by three different reasons. The paths shown in Figure 15 belong to the red markers in Figure 14b and Figure 14c. Paths 34 and 38 both got stuck for the same reason: states *BFW* and *BFT* did not find an escape point in any of the wedges. Path 14 was working its way around the trees, but got stopped by a timeout. Path 15 transitioned many times from state *BFT* to state *MtW*, and from state *MtW* back to *SB* (followed by *BFW* and *BFT*). This was because the positive direction around the obstacle boundary was forgotten by the time the transition to *SB* happened. In other words, the threshold explained in subsec-

tion 3.4 was too low for this run, since it kept going around the same group of obstacles in different directions, until the timeout was reached.

4.3 Real World

For real-world testing, two JeVois³ cameras were connected to one ARM-A7 processor, responsible for the navigation. The setup is shown in Figure 16, where the stereo camera is shielded in tin foil to block electromagnetic radiation.



Figure 16: Experimental setup: JeVois stereo camera mounted to the bottom of the drone.

The testing is performed in the CyberZoo facility at the faculty of Aerospace Engineering at the TU Delft. The drone obtains accurate position and orientation estimates via the motion capture system Optitrack. In the experiment shown in Figure 17, obstacles are placed in a V-shape to create a local minimum.



Figure 17: Real world testing environment: CyberZoo facility at the TU Delft. Obstacles are positioned to create a local minimum.

The results are shown in Figure 18, where Frustumbug has created a safe path around the obstacles. Figure 18a uses a low elevation angle to show that the height is approximately constant throughout the flight. Figure 18b shows a better perspective of the generated path. The path starts at the green dot, and the drone moves to the goal using state *MtG*. Once the obstacle is detected, the drone stops and hovers, because a waypoint is not found. State *SW* (cyan cross) starts scanning for a waypoint, and on the second left scan, a waypoint is found and the drone moves there. At both the orange crosses an obstacle (the CyberZoo edge) is detected and a new waypoint is found while in the state *MtW*. At the dark blue cross,

³<http://jevois.org/>

the waypoint is reached and state *MtG* moves the drone towards the goal. A waypoint above the obstacles was not found due to the limited vertical FoV.

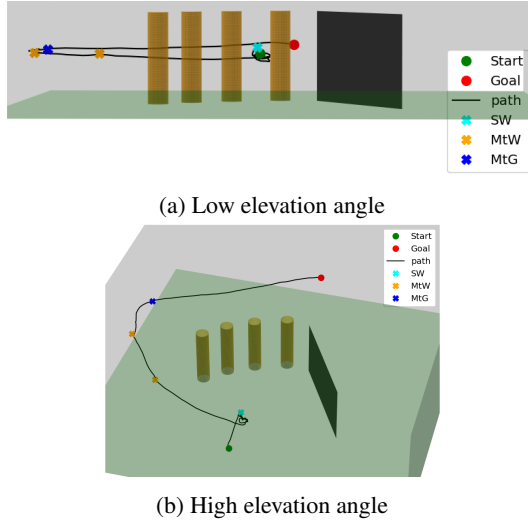


Figure 18: Result of a CyberZoo flight test. State *SW* finds a waypoint around the group of obstacles. State *MtW* steers clear of the CyberZoo edge. State *MtG* moves the drone towards the goal.

Another experiment, where the drone only had to avoid one obstacle, is shown in Figure 19. It starts at the green dot in state *MtG*, and the obstacle is detected at the orange cross. However, in this experiment, a waypoint is immediately found, hence no transition to state *SW* is necessary. State *MtW* moves the drone towards the waypoint. Upon arrival, the state transitions to *MtG* to complete the last segment to the goal. Note that these real-world tests do not test all the possible state transitions. Since the simulated environments have already proven that the algorithm can reach its goal through challenging environments, the main focus of the real-world tests is to show its feasibility on resource-limited systems.

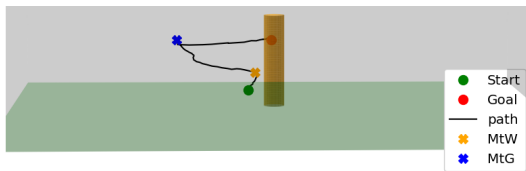


Figure 19: Result of a simple CyberZoo flight test. In state *MtG*, the obstacle is detected and a waypoint is found. State *MtW* moves the drone to the waypoint and a transition to state *MtG* follows.

5 DISCUSSION

The results have shown that Frustumbug is able to reactively plan its path from start- to goal position with an acceptable path length, using limited computation and memory.

Previous research had various simplifications and limitations. 3DBug assumed that the range sensor has perfect readings over a 360 degrees FoV and that all obstacles could be modeled as polyhedral. In addition, it was only tested in simulation. Other studies on reactive path planning in 3D did not explicitly mention escaping local minima or used randomly generated sub-optimal paths to escape. Frustumbug has been demonstrated in complex, realistic environments. It has found solutions for the assumptions that do not hold in these environments, and escapes local minima by taking logical decisions. The strengths of the proposed algorithm are the high success rate and the small size, weight and power. Furthermore, every path-planning decision is made according to a large decision tree. This means that any choice made by the algorithm can be examined afterward and changed individually. Moreover, the stereo matching parameters can be altered mid-flight if the incoming image shows signs of low texture or highly repetitive texture.

The limitations of the algorithm come in twofold, which will be discussed along with a few recommendations.

First, the choice of using a stereo camera with a small FoV limits the drone's local environment information. For example, during boundary following, the drone has to stop often to scan extra wedges. Furthermore, it results in small climbing and descending angles in states *WC* and *WD*. Increasing the FoV is a possible solution, but comes with issues. Increasing the FoV while keeping the same image resolution will decrease the focal length, which decreases the largest depth that can be measured. The stereo base could be increased to counteract, but for small drones, this is not ideal. To keep a constant focal length, the image size needs to be increased, but this will increase the computation. Another solution is to allow separate rotation of the stereo camera, similar to Wedgebug. However, this would introduce more moving components that can fail and puts more requirements on drones that would like to use Frustumbug. A third solution would be the use of additional lightweight sensors, for example, an HC-SR04 SONAR. Since it has an extremely small FoV it could not be used for normal path planning, but it could scan to see if there is an obstacle above the drone. If not, the FoV limited climbing maneuver can be avoided. Pointing a stereo camera up would be excessive since detailed information about obstacles above the drone is not required. The SONAR requires only 15 mA (JeVois: 800 mA), but adds an extra weight of 8.7 grams per sensor⁴.

Second, the c-space expansion could hide any escape points in narrow streets, or local areas with a high obstacle density. Future work could focus on running two instances of

⁴<https://www.adafruit.com/product/3942>

the c-space algorithm, with both a small and large expansion radius. The small expansion radius could be used temporarily to escape the situations described above. However, the larger expansion radius is the default for path planning, to keep a big enough safety margin from obstacles for changes caused by wind or GPS inaccuracy.

Further improvements can be found when looking at the results. The black path in Figure 10a failed because it did not identify that it was above an obstacle that needed to be passed, as described in subsection 3.5. In this particular flight, the drone was squeezed between two buildings of different heights, and no linearly increasing depth values were found in the direction of the goal. Future work could focus on designing an improved version of this function. The same rationale applies to the green path: it failed because the trick to ignore repetitive texture caused it to not recognize another building. Future work could use image characteristics to have a variable parameter set for the stereo-matching algorithm.

Looking at Figure 10b, improvements can be made in the decision for state transition. For example, state *SB* is activated when the obstacle is relatively small (and state *SW* could not find a waypoint), but once state *BFW* or *BFT* realizes that the obstacle is in fact quite large, the state should switch to state *SC* and not stay in *BFW* or *BFT*, as was seen in Figure 10b. The same figure also shows a red path getting stuck in a loop around the same building, hence implementing something equivalent to a 3D loop detection seems useful after all.

6 CONCLUSION

Frustumbug is a cheap, lightweight three-dimensional path-planning package for small drones. It reaches more than 90% of its goals in complex environments, is robust to noisy range sensor data and runs smoothly on a 20-gram stereo vision system with limited memory and computation. Furthermore, it has been extensively tested, both in simulation and real-world.

Our paper takes a step towards a publicly available path planning package. Since it does not rely on heavy processing power, it will be less demanding for the battery and onboard computer, increasing valuable flight time.

REFERENCES

- [1] N Aswini, E Krishna Kumar, and SV Uma. Uav and obstacle sensing techniques—a perspective. *International Journal of Intelligent Unmanned Systems*, 2018.
- [2] Larry Matthies, Roland Brockers, Yoshiaki Kuwata, and Stephan Weiss. Stereo vision-based obstacle avoidance for micro air vehicles using disparity space. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 3242–3249. IEEE, 2014.
- [3] Junseok Lee, Xiangyu Wu, Seung Jae Lee, and Mark W Mueller. Autonomous flight through cluttered outdoor environments using a memoryless planner. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1131–1138. IEEE, 2021.
- [4] Oualid Doukhi and Deok Jin Lee. Deep reinforcement learning for autonomous map-less navigation of a flying robot. *IEEE Access*, 10:82964–82976, 2022.
- [5] Ricardo Bedin Grando, Junior Costa de Jesus, Victor Augusto Kich, Alisson Henrique Kolling, and Paulo Lilles Jorge Drews-Jr. Double critic deep reinforcement learning for mapless 3d navigation of unmanned aerial vehicles. *Journal of Intelligent & Robotic Systems*, 104(2):1–14, 2022.
- [6] Yang Yu, Wang Tingting, Chen Long, and Zhang Weiwei. Stereo vision based obstacle avoidance strategy for quadcopter uav. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 490–494. IEEE, 2018.
- [7] Sharon L Laubach and Joel W Burdick. An autonomous sensor-based path-planner for planetary microrovers. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, volume 1, pages 347–354. IEEE, 1999.
- [8] Ishay Kamon, Ehud Rivlin, and Elon Rimon. 3dbug: A three-dimensional range-sensor based globally convergent navigation algorithm. Technical report, Computer Science Department, Technion, 1996.
- [9] Tom van Dijk. Self-supervised learning for visual obstacle avoidance. Technical report, Micro Air Vehicle Lab (MAVLab), TU Delft, mar 2020. Technical report.
- [10] Kimberly McGuire, Guido De Croon, Christophe De Wagter, Karl Tuyls, and Hilbert Kappen. Efficient optical flow and stereo vision for velocity estimation and obstacle avoidance on an autonomous pocket drone. *IEEE Robotics and Automation Letters*, 2(2):1070–1076, 2017.
- [11] Hua Liu, Rui Wang, Yuanping Xia, and Xiaoming Zhang. Improved cost computation and adaptive shape guided filter for local stereo matching of low texture stereo images. *Applied Sciences*, 10(5):1869, 2020.
- [12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.
- [13] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1):7–42, 2002.

- [14] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 807–814. IEEE, 2005.
- [15] Alexios Lyrakis. Low-cost stereo-based obstacle avoidance for small uavs using uncertainty maps. Master's thesis, Delft University of Technology, 2019.
- [16] Suyog Patil, Joseph Simon Nadar, Jimit Gada, Siddhartha Motghare, and Sujath S Nair. Comparison of various stereo vision cost aggregation methods. *International Journal of Engineering and Innovative Technology*, 2(8):222–226, 2013.
- [17] Taha Elmokadem and Andrey V Savkin. Towards fully autonomous uavs: A survey. *Sensors*, 21(18):6223, 2021.
- [18] V. Lumelsky and A. Stepanov. Dynamic path planning for a mobile automaton with limited information on the environment. *IEEE Transactions on Automatic Control*, 31(11):1058–1063, 1986. doi: 10.1109/TAC.1986.1104175.
- [19] P Rajashekar Reddy, V Amarnadh, and Mekala Bhaskar. Evaluation of stopping criterion in contour tracing algorithms. *International Journal of Computer Science and Information Technologies*, 3(3): 3888–3894, 2012.
- [20] Norlida Buniyamin, W Wan Ngah, Nohaidda Sariff, Zainuddin Mohamad, et al. A simple local path planning algorithm for autonomous mobile robots. *International journal of systems applications, Engineering & development*, 5(2):151–159, 2011.
- [21] Min Cheol Lee and Min Gyu Park. Artificial potential field based path planning for mobile robots using a virtual obstacle concept. In *Proceedings 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003)*, volume 2, pages 735–740. IEEE, 2003.
- [22] Joe Sfeir, Maarouf Saad, and Hamadou Saliah-Hassane. An improved artificial potential field approach to real-time mobile robot path planning in an unknown environment. In *2011 IEEE international symposium on robotic and sensors environments (ROSE)*, pages 208–213. IEEE, 2011.
- [23] Helen Oleynikova, Dominik Honegger, and Marc Pollefeys. Reactive avoidance using embedded stereo vision for mav flight. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 50–56. IEEE, 2015.

Improved Moving Horizon Estimation for Ultra-Wideband Localization on Small Drones

S. Pfeiffer*, C. De Wagter, and G.C.H.E. de Croon
Delft University of Technology, The Netherlands

ABSTRACT

Moving Horizon Estimation (MHE) offers multiple advantages over Kalman Filters when it comes to the localization of drones. However, due to the high computational cost, they can not be used on Micro Air Vehicles (MAVs) with limited computational power. We have previously shown, that with a few assumptions and simplifications, MHE can be made more efficient while retaining good localization performance. In this paper, we present two additional improvements: the introduction of dynamic step sizes to the gradient descent algorithm, which leads to a significant increase in robustness, and the use of switching variables for outlier rejection, which further reduces the computational load. Both improvements are implemented and assessed in simulation and experiments. Using dynamic step sizes makes it possible to reliably use the estimator on board of a real drone, and the use of Newton's method specifically opens the option to add different types of measurements. The new outlier rejection method on the other hand is shown to reduce the computational load significantly while having no big impact on accuracy.

1 INTRODUCTION

Self-localization is an important task for autonomous drones, especially when operating in indoor environments. Since navigation using Global Navigation Satellite Systems (GNSS) exhibits a significant decrease in performance when used inside of buildings, other solutions are needed. Ultra-Wideband (UWB) technology has been investigated as a promising alternative in scenarios where the environment can be prepared in advance. UWB localization works in a similar fashion as GNSS, but instead of ranging with respect to satellites, it uses fixed beacons at known locations in the environment [1].

The UWB range measurements can be used for triangulation [2], but are more commonly fused with measurements from an inertial measurement unit (IMU) for more accurate results [3]. Kalman filter variants such as the Extended Kalman Filter (EKF) are often used due to their speed and

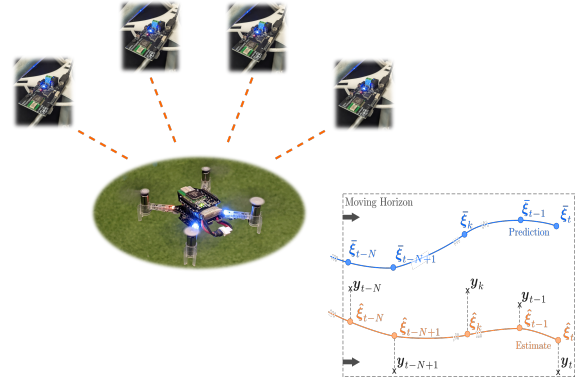


Figure 1: Moving Horizon Estimators (MHEs) usually requires a lot of computational power. Improvements to our previous work on computationally efficient MHEs allow for on-board execution of the algorithm to localize a nano-copter using Ultra-Wideband measurements.

efficiency. However, they are not ideal when working with highly non-linear dynamics or non-Gaussian noise, both of which are present in the system in question [4].

Moving Horizon Estimators (MHEs) are well suited for state estimation in non-linear systems with non-Gaussian noise, but are computationally expensive, which makes them impractical to use on small drones with limited computational power. In our previous work, we have investigated several simplifications that reduce the load for localization of small drones using an MHE [4]. While the resulting estimator performed well in offline evaluations on real measurement data, its robustness was found to be lacking. Specifically in situations where the number of incoming measurements varies a lot, the tuning had to be quite conservative to avoid unstable behaviour when many measurements were coming in. Furthermore, while we were happy with the performance of Random Sample Consensus (RANSAC) for outlier rejection, its computational cost was quite high for an algorithm aimed at computationally extremely limited systems, like the micro-controllers on board of nano-drones.

In this paper we will present the following two improvements to efficient moving horizon estimation for localization with UWB:

- Using a dynamic step size in the gradient-descent step to improve robustness

*Email address(es): s.u.pfeiffer@tudelft.nl

- Outlier rejection based on switching variables to further improve the computational efficiency

2 PRELIMINARIES

2.1 Computationally Efficient MHE

Let us quickly review the computationally efficient MHE presented in [4] to highlight some of its shortfalls. We again consider the discrete-time non-linear dynamic system given by (1), expressed in terms of the system state ξ , input vector u , measurement vector y , process noise w and measurement noise v , all at timestep t respectively.

$$\begin{cases} \xi_{t+1} = f(\xi_t, u_t) + w_t \\ y_t = h(\xi_t, u_t) + v_t \end{cases} \quad (1)$$

The full MHE problem for a horizon of length $N + 1$ is given by the optimization problem (2), formulated in terms of the decision variables ξ , w and v . Put into words, the MHE tries to minimize the squared observed process and measurement noise, as well as changes to the prior estimate (also called the arrival cost). The prior $\bar{\xi}_{t-N}$ is calculated from the previous estimate using the prediction equation, $\bar{\xi}_{t-N} = f(\bar{\xi}_{t-N-1})$.

$$\begin{aligned} \min_{\xi, w, v} \quad & \mu \|\xi_{t-N} - \bar{\xi}_{t-N}\|^2 + \sum_{k=t-N}^{t-1} w_k^2 + \sum_{k=t-N}^t v_k^2 \\ \text{s.t.} \quad & \xi_{k+1} = f(\xi_k, u_k) + w_k \\ & y_k = h(\xi_k, u_k) + v_k \end{aligned} \quad (2)$$

By assuming negligible process noise ($w = 0$), the problem can be reduced to an output error MHE. Furthermore, by defining the composite prediction and measurement equations $\mathcal{F}_k(\xi_{t-N})$ and $\mathcal{H}_{i,k}(\xi_{t-N})$, we can merge the constraints into the cost-function J , which now only depends on the decision variable ξ_{t-N} . Note that the number of measurements m_t can differ between timesteps.

$$\mathcal{F}_k(\xi_{t-N}) = f^{u_k} \circ \dots \circ f^{u_{t-N}}(\xi_{t-N}) \quad (3)$$

$$\mathcal{H}_{i,k}(\xi_{t-N}) = h_i^{u_k} \circ \mathcal{F}_{k-1}(\xi_{t-N}) \quad (4)$$

$$\begin{aligned} J(\xi_{t-N}) = & \mu \|\xi_{t-N} - \bar{\xi}_{t-N}\|^2 \\ & + \sum_{k=t-N}^t \sum_{i=1}^{m_t} (\|y_{i,k} - \mathcal{H}_{i,k}(\xi_{t-N})\|^2) \end{aligned} \quad (5)$$

To save computation time, instead of fully solving the optimization problem at every timestep, only a single gradient descent step is performed as suggested by [5].

$$\hat{\xi}_{t-N} = \bar{\xi}_{t-N} - \alpha \nabla J(\bar{\xi}_{t-N}) \quad (6)$$

Using the chain rule, the gradient of the cost function is easily calculated analytically. When using only a single iteration, the gradient is calculated with the prior best estimate

$\xi_{t-N} = \bar{\xi}_{t-N}$, thus the arrival cost term is zero.

$$\nabla J(\bar{\xi}_{t-N}) = -2 \sum_{k=t-N}^t \sum_{i=1}^{m_t} [(y_{i,k} - \mathcal{H}_{i,k}(\bar{\xi}_{t-N})) \nabla \mathcal{H}_{i,k}(\bar{\xi}_{t-N})] \quad (7)$$

To save computational power, the gradient of the composite measurement equation can be simplified by iterative computation of the Jacobian of the composite prediction equation (chain rule):

$$\nabla \mathcal{H}_{i,k}(\bar{\xi}_{t-N}) = (D\mathcal{F}_{k-1}(\bar{\xi}_{t-N}))^T \cdot \nabla h_i(\bar{\xi}_k, u_k) \quad (8)$$

$$D\mathcal{F}_{k-1}(\bar{\xi}_{t-N}) = Df(\bar{\xi}_{k-1}, u_{k-1}) \cdot D\mathcal{F}_{k-2}(\bar{\xi}_{t-N}) \quad (9)$$

For outlier rejection, a RANSAC scheme was used, which can identify outliers very well but requires a significant amount of computational power.

2.2 Drone and Measurement Model

We use the drone model in [3], but represent the attitude in quaternions and simplify the drag forces. For the use in our estimator, we discretize the model using the forward Euler method.

$$\dot{x} = \left[q \otimes \begin{bmatrix} 0 \\ \rho \end{bmatrix} \otimes q^{-1} \right]_v \quad (10)$$

$$\dot{\rho} = \frac{f}{m} e_3 + (K_a - [[\omega \times]]) \rho - \left[q^{-1} \otimes \begin{bmatrix} 0 \\ g \end{bmatrix} \otimes q \right]_v \quad (11)$$

The six dimensional state vector $\xi = [x^T, \rho^T]^T$ contains the position $x \in \mathbb{R}^3$ in the global frame and the drone's velocities $\rho \in \mathbb{R}^3$ in the body frame. The system inputs are the total thrust f , the attitude quaternion $q \in \mathbb{R}^4$, and the angular velocities measured by the IMU, $\omega \in \mathbb{R}^3$. m is the drone's mass and $K_a = \text{diag}(k_\perp, k_\perp, k_\parallel)$ are the linear drag coefficients. The quaternion product is noted as $\otimes$, while the subscript 0 and v refer to the scalar and vector portion of the quaternion respectively. Finally, g is the acceleration due to gravity and $[[\omega \times]]$ is the skew-symmetric matrix defined as

$$[[\omega \times]] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \quad (12)$$

We consider two different types of UWB measurements. Two-way ranging (TWR) can be used to determine the distance between the drone and a fixed beacon located at p_A . Time-difference of Arrival (TDOA) measurements on the other hand represent the difference in the distance of the drone to two fixed beacons at positions p_A and p_B . The measurement noise is represented by η_{TWR} and η_{TDOA} respectively.

$$y_{\text{TWR}} = \|p_A - x\| + \eta_{\text{TWR}} \quad (13)$$

$$y_{\text{TDOA}} = \|p_A - x\| - \|p_B - x\| + \eta_{\text{TDOA}} \quad (14)$$

3 DYNAMIC STEP SIZE

A serious problem with the approach in [4] is the fixed step size used in the gradient descent algorithm. Specifically in situations where the numbers of measurements m_t varies between steps, the fact that the gradient of the cost function is the sum of m_t individual measurement terms can lead to large variations in the gradient's magnitude. As a result, a value for α that is optimal with 4 UWB beacons, might cause the estimator to become unstable for 8 beacons. A solution to this problem is to dynamically adjust the step size of the gradient descent algorithm. We investigated two options for choosing the step size dynamically: linesearch and Newton's method. While robustness is of course an important property of any state estimation algorithm, it should be noted that we are sacrificing computational efficiency to achieve it.

3.1 Linesearch

With the linesearch method, different values for the step size α are tested at each time step. The step size resulting in the estimate with the lowest cost is then retained for the next step. The algorithm is laid out in Algorithm 1.

Algorithm 1 Linesearch Method

```

 $t \leftarrow 0$    $\xi \leftarrow \xi_0$    $\alpha \leftarrow \alpha_0$ 
 $s \leftarrow \{0.5, 0.8, 1.0, 1.25, 2.0\}$ 
while running do
   $t \leftarrow t + 1$ 
  calculate  $\nabla J(\xi_{t-N})$ 
  for  $s_i \in s$  do
     $\xi_i = \xi_{t-N} - s_i \cdot \alpha \nabla J(\xi_{t-N})$ 
    calculate  $J_i(\xi_i)$ 
  end for
   $i_{\text{opt}} \leftarrow \text{argmin}_i(J_i)$ 
   $\xi_{t-N} \leftarrow \xi_{i_{\text{opt}}}$ 
   $\alpha \leftarrow s_{i_{\text{opt}}} \cdot \alpha$ 
end while

```

The linesearch method is easy to implement in the existing algorithm, but has the downside of scaling the step size for all state variables at once. If different types of measurements are available, this can lead to problems. As an example, if a lot of velocity estimates are available but only few position estimates, it would be good to choose different step sizes for the different states. An approach that allows for this flexibility is the use of Newton's method.

3.2 Newton's Method

Newton's method is more complex to implement than linesearch, but offers two additional advantages. First, there is no need to calculate the cost of multiple estimates (which requires several predictions through the complete horizon) and second, Newton's method allows to adjust the step size along different axes. Specifically, instead of using a scalar step size α , the gradient of the cost function is multiplied by

the inverse Hessian matrix of the cost function:

$$\hat{\xi}_{t-N} = \bar{\xi}_{t-N} - (D^2 J(\bar{\xi}_{t-N}))^{-1} \nabla J(\bar{\xi}_{t-N}) \quad (15)$$

The Hessian matrix of the cost function can be obtained by calculating the Jacobian of the cost function gradient.

$$D^2 J(\bar{\xi}_{t-N}) = 2\mu\mathcal{I} - 2 \sum_{k=t-N}^t \sum_{i=0}^{m_t} (y_{i,k} D^2 \mathcal{H}_{i,k}(\bar{\xi}_{t-N}) - \nabla \mathcal{H}_{i,k}(\bar{\xi}_{t-N}) \nabla \mathcal{H}_{i,k}(\bar{\xi}_{t-N})^T - \mathcal{H}_{i,k}(\bar{\xi}_{t-N}) D^2 \mathcal{H}_{i,k}(\bar{\xi}_{t-N})) \quad (16)$$

As with the gradients, the Hessian matrices of the cost and measurement functions can be calculated algebraically in an iterative manner while stepping through the horizon. Making use of the chain rule, we find the following expression [6]:

$$D^2 \mathcal{H}_{i,k}(\xi) = (D\mathcal{F}_{k-1}(\xi))^T \cdot D^2 h(\mathcal{F}_{k-1}(\xi)) \cdot D\mathcal{F}_{k-1}(\xi) + \sum_{j=1}^n \frac{\partial h}{\partial \xi_j}(\mathcal{F}_{k-1}(\xi)) \cdot D^2 \mathcal{F}_{k-1}^j(\xi) \quad (17)$$

Luckily, many of the terms appearing in this equation are already needed when calculating the cost function gradient. Furthermore, since the prediction model we use is linear in terms of the states, the Hessian matrix of the composite prediction equations turns out to be zero. The simplified expression for the Hessian matrix of the cost function thus only depends on the Hessian matrix of the measurement function and the Jacobian matrix of the composite prediction equation:

$$D^2 \mathcal{H}_{i,k}(\xi) = (D\mathcal{F}_{k-1}(\xi))^T \cdot D^2 h(\mathcal{F}_{k-1}(\xi)) \cdot D\mathcal{F}_{k-1}(\xi) \quad (18)$$

4 SWITCHING VARIABLE OUTLIER REJECTION

Since using RANSAC for outlier rejection has the disadvantage of being computationally heavy, we decided to look for a more efficient method. The need for outlier rejection in optimization problems also arises in Simultaneous Localization and Mapping (SLAM), where incorrectly identified loop-closures can cause large errors in the resulting map. A possible approach to deal with this issue is the addition of switching variables that multiply the constraints in question [7]. Applying this concept to our MHE, we introduce for each measurement $y_{i,k}$ an associated switching variable $s_{i,k}$. By giving the optimization algorithm the ability to change these switching variables, the algorithm can enable and disable measurements based on how they impact the value of the cost function.

To use the switching variables as decision variables, they must be continuous. At the same time, for their function to enable and disable measurements, binary values would be more useful. Following [7], we use a sigmoid function to convert the continuous switching variables into values between 0 and 1, which then multiply the measurement equation. A simple sigmoid function to use is the logistic function (19).

$$\text{sig}(s_{i,k}) = \frac{1}{1 + e^{-s_{i,k}}} \quad (19)$$

In the context of calculating gradients, it is also helpful to note the derivative of the logistic function (20).

$$\frac{\partial}{\partial s_{i,k}} \text{sig}(s_{i,k}) = \text{sig}(s_{i,k}) (1 - \text{sig}(s_{i,k})) \quad (20)$$

It is important to penalize the estimator for ignoring a measurement. We do this by adding a term similar to the arrival cost, that penalizes switching variables that deviate from a prior value $\bar{s}$, which enables the measurement (e.g. $\bar{s} = 5$). To simplify notation, we collect all switching variables in a single column vector $\mathbf{s} = [s_{i,j}]$. The resulting cost function including switching variables then looks like this:

$$\begin{aligned} J(\xi_{t-N}, \mathbf{s}) = & \mu \|\xi_{t-N} - \bar{\xi}_{t-N}\|^2 \\ & + \mu_s \cdot \sum_{k=t-N}^t \sum_{i=1}^m (s_{i,k} - \bar{s})^2 \\ & + \sum_{k=t-N}^t \sum_{i=1}^m (\text{sig}(s_{i,k}) \cdot \|y_{i,k} - \mathcal{H}_{i,k}(\xi_{t-N})\|^2) \end{aligned} \quad (21)$$

When calculating the gradient of the cost function, we now also need to consider the derivative with respect to the individual switching variables. Luckily, each switching variable only appears once in each of the two sums and the corresponding derivative is easily calculated:

$$\begin{aligned} \frac{\partial J}{\partial s_{i,k}}(\xi_{t-N}, \mathbf{s}) = & 2\mu_s (s_{i,k} - \bar{s}) \\ & + \text{sig}(s_{i,k}) (1 - \text{sig}(s_{i,k})) \cdot \|y_{i,k} - \mathcal{H}_{i,k}(\xi_{t-N})\|^2 \end{aligned} \quad (22)$$

The changes to the derivatives with respect to ξ_{t-N} are also small, one only has to add the sigmoid multiplication to each of the terms of the sums:

$$\begin{aligned} \frac{\partial J}{\partial \xi_{t-N}}(\xi_{t-N}, \mathbf{s}) = & -2 \sum_{k=t-N}^t \sum_{i=1}^m [\text{sig}(s_{i,k}) (y_{i,k} - \mathcal{H}_{i,k}(\bar{\xi}_{t-N})) \nabla \mathcal{H}_{i,k}(\bar{\xi}_{t-N})] \end{aligned} \quad (23)$$

The full gradient could now be formed from the partial derivatives with respect to the state and the individual switching variables, but since the switching variables are decoupled from the rest of the system, it makes sense to update them individually using Newton's method:

$$\hat{s}_{i,k} = \hat{s}_{i,k-1} - \left(\frac{\partial^2}{\partial s_{i,k}^2} J(\bar{\xi}_{t-N}, \hat{\mathbf{s}}) \right)^{-1} \frac{\partial}{\partial s_{i,k}} J(\bar{\xi}_{t-N}, \hat{\mathbf{s}}) \quad (24)$$

For large outliers, even with the use of a switching penalty, the switching variable can take very large negative values, resulting in a variable overflow for the intermediate result of $e^{-s_{i,k}}$. To avoid this issue, it makes sense to place a lower limit on the value of $s_{i,k}$.

There is one final consideration regarding switching outlier rejection, which is the initial value of the switching variable. For simplicity, we decided to use the Mahalanobis distance based on only the measurement noise (the MHE doesn't estimate state covariance) of the associated measurement.

5 OFFLINE EVALUATION

The presented improvements were evaluated offline, using real UWB measurements gathered in the "Cyberzoo" test arena at TU Delft [8]. We simulated the performance on 12 different flights for TWR and 9 different flights for TDOA. The simulations were performed 10 times with different seeds for the random number generator, for a total of 120 simulations for TWR and 90 simulations for TDOA. For every simulation, all estimators were executed in parallel and saw therefore the same noise. For comparison, we also simulated the initial version of the MHE using the simple gradient (SG) method with a fixed step size and RANSAC for outlier rejection from [4], as well as the EKF from [3].

We compared the different estimators based on three metrics, which are presented in Figure 2 for TWR and in Figure 3 for TDOA.

- **Success rate:** We defined a run to be successful if the root-mean-square error (RMSE) on the position stays below 0.5 m. This definition is somewhat arbitrary but is meant to reflect a value at which the estimator's accuracy is good enough to allow navigation in spacious indoor environments.
- **Accuracy:** The accuracy of the estimators is assessed using the position RMSE. Unsuccessful runs were removed from the data prior to plotting the accuracy statistics, since unsuccessful runs for only a few beacons can easily reach errors of tens of meters and therefore have a tendency to distort the plots.
- **Normalized computation time:** To assess the computational efficiency, we calculated the average computation time per step. For every run, we normalize the average by dividing by the average computation time of the EKF. This was done to reduce the effect of the hardware used and the system load from other sources.

At first glance we can see, that TWR yields more stable results. The success rate of all estimators is near identical and reaches 100% at 5 or 6 beacons. Looking at the accuracy, we can see that all MHE versions outperform the EKF slightly, but no difference can be seen between the different versions.

The TDOA results are a bit messier. The success rate is quite different between different versions and for many estimators never reaches 100%. It is therefore more interesting to compare the quality of the estimates based on the success rate, rather than based on the accuracy of the successful runs. All MHE variants surpass the EKF in terms of success rate

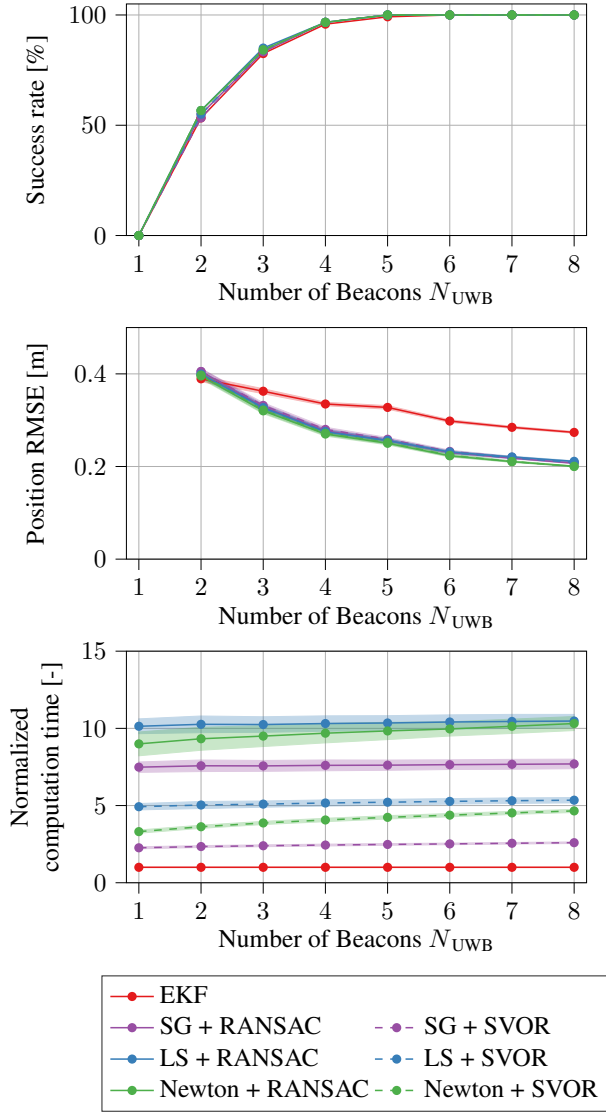


Figure 2: Performance of MHE configurations in simulation using TWR measurements: Simple gradient (SG), line-search (LS) and Newton's method to solve the minimization problem, random sample consensus (RANSAC) and switching variable outlier rejection (SVOR) for outlier rejection. Shaded regions represent the standard error of the mean.

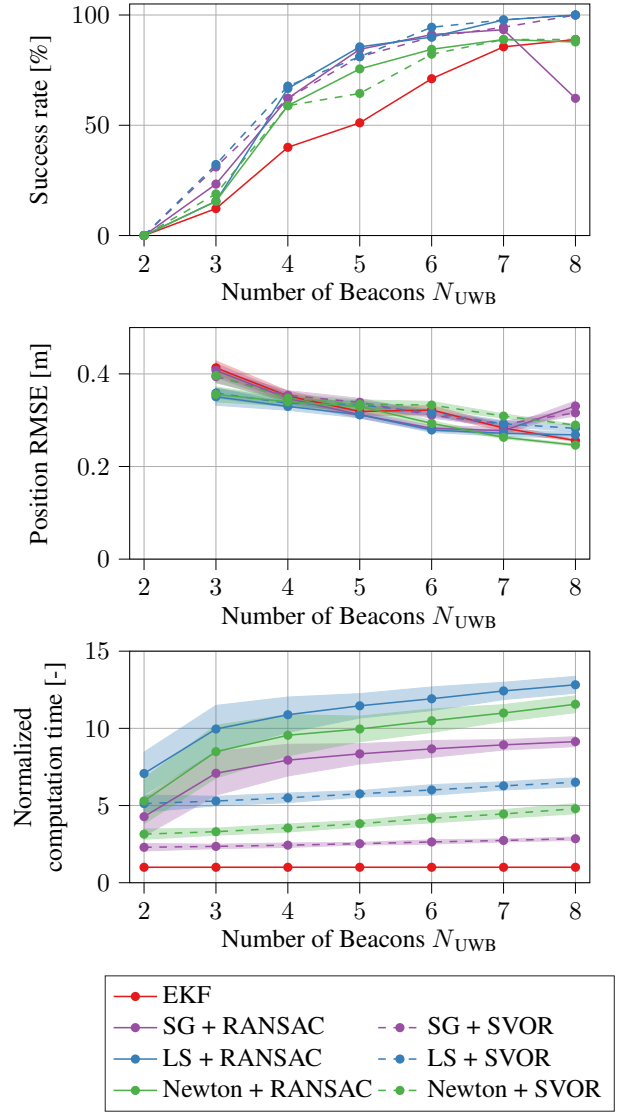


Figure 3: Performance of MHE configurations in simulation using TDOA measurements: Simple gradient (SG), line-search (LS) and Newton's method to solve the minimization problem, random sample consensus (RANSAC) and switching variable outlier rejection (SVOR) for outlier rejection. Shaded regions represent the standard error of the mean.

up to 7 beacons. The effect of a constant step size for varying numbers of measurements can be seen in the decreased performance of the simple gradient MHE when the number of beacons increases to 8. Interestingly, using switching variables for outlier rejection (SVOR) seems to be another suitable method for reducing this problem, as seen in the higher success rate. This is likely due to the fact, that SVOR can turn off measurements to keep the magnitude of the gradient closer to constant. This does however mean, that the estimator does not profit from the increased number of measurements, which becomes apparent when looking at the accuracy plots. The most successful approach for dealing with variable gradient magnitudes seems to be the linesearch (LS) approach. We suspect however, that in scenarios that incorporate other measurements (e.g. velocity from optic flow), using Newton's method will be advantageous.

Let's finally have a look at the computational efficiency. The EKF clearly still outperforms all versions of the MHE, but as expected, SVOR clearly reduces the computation time compared to RANSAC. Looking at the methods for choosing dynamic step sizes, Newton's method is more efficient than Linesearch, but still adds some additional cost compared to the simple gradient method.

6 ON-BOARD EXPERIMENTS

To verify our simulation results and demonstrate that the MHE is indeed capable of running on computationally limited devices, we implemented the code on a Crazyflie 2.1. The Crazyflie 2.1. is a small (38 g), versatile platform that can be equipped with a variety of expansion boards, such as the UWB Deck for UWB ranging and the Flow Deck, for altitude and flow measurements. To make full use of the Flow Deck and challenge the versatility of our estimator, we not only used the time-of-flight measurements for altitude, but also incorporated the measurement equations for flow and adjusted the MHE formulation to enable the use of measurements with different units (the flow measurements are in pixels per second). Specifically, we use Newton's method with SVOR and introduce the inverse covariance matrix of the prior, P^{-1} , and the inverse measurement variance, σ^2 as weights in the cost function [9]:

$$\begin{aligned}
 J(\xi_{t-N}, s) = & \mu (\xi_{t-N} - \bar{\xi}_{t-N})^T P^{-1} (\xi_{t-N} - \bar{\xi}_{t-N}) \\
 & + \mu_s \cdot \sum_{k=t-N}^t \sum_{i=1}^m (s_{i,k} - \bar{s})^2 \\
 & + \sum_{k=t-N}^t \sum_{i=1}^m \left(\text{sig}(s_{i,k}) \cdot \frac{\|y_{i,k} - \mathcal{H}_{i,k}(\xi_{t-N})\|^2}{\sigma^2} \right)
 \end{aligned} \quad (25)$$

This way, the units of the measurement errors are removed and the measurements are immediately weighted according to their reliability. Each estimator was flown at least five times for a given number of beacons (4, 6 and 8) with the UWB ranging set to TWR. The results in Figure 4 show, that the MHE is indeed able to compete with the EKF in terms

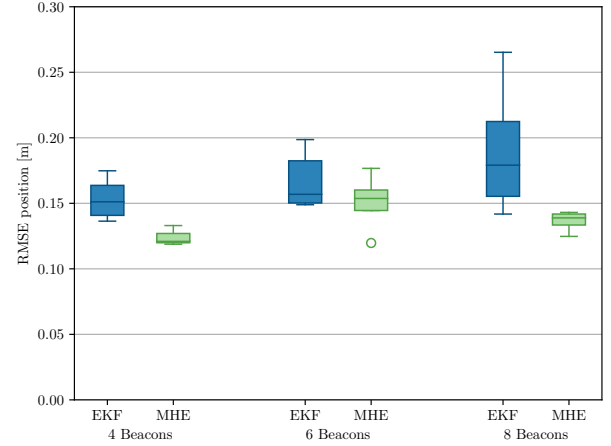


Figure 4: Comparison of MHE and EKF using Two-Way Ranging (5 flights per setting)

of performance for the given scenarios. In terms of computational requirements, we limited the number of measurements the MHE keeps in memory to 80 and used a quality measure based on the switching variable to remove bad measurements before they fall out of the window. With these changes, the MHE runs on-board, using about 25% of the available computational resources, whereas the EKF operates with only about one-fifth of this number. This actually matches quite well with the observations from the offline evaluation.

7 CONCLUSION

Moving Horizon Estimation holds a lot of potential, even for computationally limited platforms. By simplifying the problem to an output error MHE and approximating the optimization problem with a single-step gradient descent algorithm, the computational requirements of MHE can be reduced significantly, albeit at the cost of a bit of accuracy. In the end, the MHE slightly outperforms the extended Kalman Filter, which is a popular option for any non-linear state estimation problem.

The use of dynamic step sizes in a single-step gradient descent algorithm helps alleviate problems that occur when the number of incoming measurements varies greatly. Both a linesearch approach and Newton's method are suitable, but Newton's method offers more flexibility when dealing with different kinds of measurements.

To further reduce the computational cost, we employed outlier rejection based on switching variables, which offers advantages over RANSAC in terms of computation time.

The improved MHE shows promise in simulation and experiments, where it outperforms a classical EKF and previous iterations of our computationally efficient MHE. The main limitations of the approach are the more complex formulation

and the still non-negligible increase in computational cost.

In its current formulation, the MHE only estimates position and velocity, while attitude estimates are provided by the complementary filter. This approach is suitable for pitch and roll, since the gravitational pull can be used as reference. The yaw estimates on the other hand are subject to drift and currently require proper initialization. This problem should be addressed in future work by including yaw (error) as an additional state in the model.

REFERENCES

- [1] Abdulrahman Alarifi, Abdulmalik Al-Salman, Mansour Alsaleh, Ahmad Alnafessah, Suheer Al-Hadhrami, Mai A. Al-Ammar, and Hend S. Al-Khalifa. Ultra wide-band indoor positioning technologies: Analysis and recent advances. *Sensors (Switzerland)*, 16(5):1–36, 2016.
- [2] Abdelmoumen Norrdine. An Algebraic Solution to the Multilateration Problem. In *International Conference on Indoor Positioning and Indoor Navigation*, 2012.
- [3] Mark W. Mueller, Michael Hamer, and Raffaello D’Andrea. Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadcopter state estimation. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1730–1736. IEEE, 2015.
- [4] Sven Pfeiffer, Christophe De Wagter, and Guido C.H.E. de Croon. A computationally efficient moving horizon estimator for ultra-wideband localization on small quadrotors. *IEEE Robotics and Automation Letters*, 6(4):6725–6732, 2021.
- [5] Angelo Alessandri and Mauro Gaggero. Fast moving horizon state estimation for discrete-time systems using single and multi iteration descent methods. *IEEE Transactions on Automatic Control*, 62(9):4499–4511, 2017.
- [6] Maciej Skorski. Chain rules for hessian and higher derivatives made easy by tensor calculus, 2019.
- [7] Niko Sünderhauf and Peter Protzel. Towards a robust back-end for pose graph SLAM. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1254–1261, 2012.
- [8] Sven Pfeiffer, Christophe De Wagter, and Guido de Croon. Data underlying the publication: ”A Computationally Efficient Moving Horizon Estimator for UWB Localization on Small Quadrotors”. 6 2021.
- [9] Christopher V Rao. *Moving horizon strategies for the constrained monitoring and control of nonlinear discrete-time systems*. The University of Wisconsin-Madison, 2000.

Efficient Model-Aided Visual-Inertial Ego-Motion Estimation for Multirotor MAVs

Yingfu Xu and Guido C. H. E. de Croon *
Delft University of Technology

ABSTRACT

When deployed onboard micro air vehicles (MAVs) with limited processing power, visual ego-motion estimation solutions face an efficiency-accuracy trade-off. This paper proposes an aerodynamic-model-aided approach that emphasizes time efficiency over estimation accuracy. A linear drag force model of propellers guarantees bounded estimation errors in the velocity components orthogonal to the shafts of propellers and the attitude relative to the gravity direction. Feature point correspondences are extracted from the monocular image stream to compute the relative heading angle and translational direction, which is fused with inertial measurements by an extended Kalman filter (EKF) in a loosely coupled manner. The proposed approach shows balanced performance in accuracy and efficiency. It also has robustness to situations where vision information becomes unavailable.

1 INTRODUCTION

The autonomous flight of micro air vehicles (MAVs) in GPS-denied environments is challenging. For large flying robots (*i.e.* heavier than ~ 300 grams, and larger than ~ 50 centimeters in diameter), this task is substantially simpler since they are less constraint by computational or payload capacity, and can process information from numerous sensors. However, they require more flying space and stringent safety checks, and are generally more expensive. In contrast, palm-sized or smaller MAVs [1] are promising alternatives as being safer and cheaper. But they come with the disadvantage of limited sensing and processing resources. It makes monocular visual-inertial odometry (VIO) that combines a single camera and an inertial measurement unit (IMU) the most promising option for ego-motion estimation. The camera provides abundant up-to-scale information about the surroundings, and the IMU measures metric-scale rotational rate and translational acceleration in high frequency.

VIO [2, 3, 4, 5] has shown good performance in MAV navigation in recent years. Ego-motion estimation and environment mapping can be performed simultaneously by taking the reprojection error from observations of the same landmark

in different frames and the integrated IMU measurements as constraints between the camera poses and landmarks' locations. The locations of landmarks are treated as states to be estimated, along with camera poses. Environment mapping not only contributes to the accuracy of the ego-motion estimation, but also provides an obstacle map for motion planning. However, this comes at the cost of iterative computation and the need for an initialization procedure, which may be required multiple times in case of tracking failure. Even for approaches [3] that map sparse-point feature points and use a sliding window to bound the optimization, the computational demand of this task is still considerable.

Focusing on ego-motion, the multi-state constraint Kalman filter (MSCKF) [2] maintains a window of camera poses but does not optimize landmark location. It is based on an extended Kalman filter (EKF) whose state vector is augmented with poses of the previous frames that observed the same feature points. The least-squares location of a point is only calculated once after it is no longer visually tracked. Its reprojection errors, which express geometric constraints between the camera poses, are then used in visual measurement updates. Instead of reprojecting 3D point locations into multiple frames, [5] enforces multiple constraints on the relative pose of a frame pair through the two observations of the same point. The drift in the relative pose is reflected by the residual of the epipolar geometry constraint. This residual updates the paired camera poses in an EKF. The absence of point locations reduces computational demand, and only one frame's pose is needed to augment the state vector. However, point correspondences perform updating one by one using the raw pixel locations of tracked points. The estimation accuracy can suffer from frame-to-frame feature tracking noise.

The above-mentioned approaches use the pixel locations of feature points. Visual and inertial measurements are fused in a tightly-coupled manner. Visual odometry (VO) systems like the SVO [6] produce up-to-scale environment map and ego-motion estimation that can be loosely coupled with IMU measurements by an EKF [7]. The EKF-based, loosely-coupled approach SVO+MSF [8] and the EKF-based, tightly-coupled, map-less MSCKF outperform others in efficiency among several VIO solutions [9].

In windless environments, since all the aerodynamic force acting on an MAV is caused by propeller rotation and ego-motion, the aerodynamic model can be an information source for ego-motion estimation. As shown in [10], a simplified linear drag model is combined with IMU measurements to es-

*Email addresses: yingfu.xu.94@gmail.com, g.c.h.e.decroon@tudelft.nl

time horizontal components of attitude and velocity in the multirotor's body frame. Unlike estimation from purely integrated IMU measurements, the error of this model-aided estimation does not increase over time and is related to IMU bias and model fidelity. This aerodynamic model was used for velocity prediction [1, 5] and trajectory tracking [11]. In [12], high estimation accuracy was reached with a more precise dynamic model that takes into account thrust forces and the effect of rotor speed on the drag, which makes the vertical speed in the body frame to be observable. The camera provides point correspondences only to estimate the relative yaw angle between a pair of frames. Worth noticing is that the drift-free roll and pitch angles were used as known values to simplify the calculation of the relative pose.

To gain robustness and accuracy with as little computation as possible, we propose an approach that combines all the previously mentioned strategies that benefit time efficiency. It is a map-less, model-aided, EKF-based, loosely-coupled ego-motion estimator for multirotor MAVs. The linear drag force model prevents attitude and velocity estimation in the body's horizontal plane from drifting over time when there is no visual information. The relative heading angle and the direction of translational motion between two frames are calculated from visual feature point correspondences using the epipolar constraint. The attitude estimation is taken as known information to simplify the visual pose calculation. The visual update executes in a one-frame-one-time manner.

The proposed ego-motion estimator has a relatively low-complexity modular pipeline that is easy to implement and debug. We choose the EuRoC [13] dataset as the validation tool and compare our approach with MSCKF [2], a relatively efficient (yet accurate) VIO solution. The accuracy of the proposed approach is compromised due to the prioritized efficiency. But it is sufficient for short-time navigation. And it is observed that the estimator can maintain its accuracy when visual information is no longer available.

2 ESTIMATOR FRAMEWORK

The proposed ego-motion estimator pipeline is shown in Fig. 1. The *Visual Processing* module receives estimated roll and pitch angles from the *EKF* module. The relative pose is calculated from the feature point correspondences in the current frame and the keyframe based on the estimated roll and pitch angles. The relative yaw angle and the direction of translation with respect to the keyframe are taken as measurements for the visual update step of the EKF.

2.1 Definitions

In this paper, we denote all scalars by lowercase letters x , vectors by lowercase bold letters $\mathbf{x}$, and matrices by bold uppercase letters $\mathbf{X}$. Coordinate frames are denoted by non-bold uppercase letters X . Estimated values are written as $\hat{x}$, and raw measurements as $\tilde{x}$.

As shown in Fig. 2, the body frame B (green) is defined according to the pose of the propellers. The plane defined

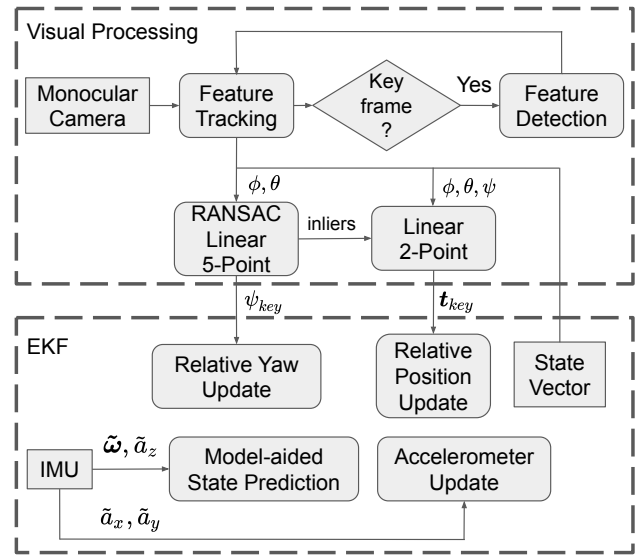


Figure 1: Pipeline of the proposed approach.

by the x -axis and y -axis of B is orthogonal to the propellers' rotation axis. The IMU is located at the origin of B . IMU measurements are required to be expressed in B before being used by the estimator. The world frame W (black) is stationary. Its z -axis z_W has the same direction as gravity. The heading frame H (red) is defined by rotating the world frame around the z -axis by the yaw angle. Thus z_H always points to the direction of gravity. The heading frame's origin coincides with the origin of the camera frame C . The Euler angles roll ϕ , pitch θ , and yaw ψ reflect the rotation between W and B . The rotation sequence is first z -axis (ψ), then y -axis (θ), and last x -axis (ϕ). ϕ and θ reflect the attitude of B respect to the gravity direction.

If the number of tracked feature points is below the pre-set threshold, or there are not enough inlier points in the RANSAC-based calculation of the essential matrix, the current frame is defined as the new keyframe. A certain number of feature points are detected in a new keyframe. Unit vector t_{key} points from the camera position of the keyframe (right) to the current camera position (left), as shown in Fig. 2. The solid lines connect the camera with the feature points detected in the keyframe. The dashed lines connect the camera with points tracked in the current frame.

For cameras mounted far from the body center, we consider the translational velocity of the camera caused by its rotation relative to IMU, which is neglected in [5, 12]. The vector that points from the IMU to the camera expressed in the body frame is denoted by p_B^C .

2.2 Linear Drag Model

There are various aerodynamic forces acting on an multirotor MAV in flight. In [12], the thrust produced is modelled to be proportional to the sum of the squares of pro-

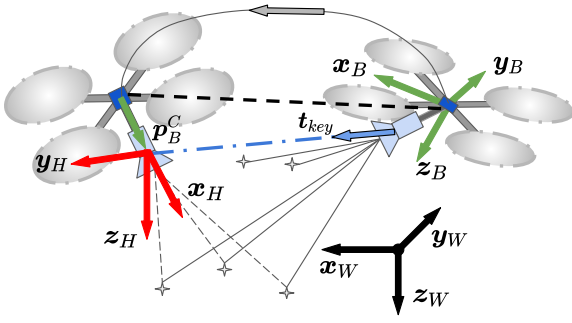


Figure 2: Schematic that illustrates the definition of the different coordinate frames and the motion of the MAV from the key frame pose to the current pose.

pellor speed. The drag acting on propellers is proportional to the product of the MAV's velocity and the sum of the rotor speeds. The velocity-square-proportional parasitic drag on the airframe was considered in [14] to improve the accuracy of the model in high-speed flight.

With the sensor setup that only consists of an IMU and a camera, we choose the simplified velocity-proportional drag force model [10]. We define the propeller plane as the plane orthogonal to the shafts of propellers. The drag force vector inside the propeller plane is approximately proportional to the projection of the MAV's velocity vector to the propeller plane. The drag parameter k_d is estimated as an EKF state to compensate for its variation in different aerodynamic regimes. Its derivative is modeled as a small Gaussian white noise w_{k_d} .

The biased and noisy IMU measurements are modeled as

$$\tilde{\mathbf{a}} = \hat{\mathbf{a}} + \mathbf{b}_a + \mathbf{w}_a, \quad \tilde{\boldsymbol{\omega}} = \hat{\boldsymbol{\omega}} + \mathbf{b}_g + \mathbf{w}_g, \quad (1)$$

where $\mathbf{w}_a$ and $\mathbf{w}_g$ are white Gaussian noise. We model the derivatives of the additive accelerometer bias $\mathbf{b}_a$ and gyroscope bias $\mathbf{b}_g$ as small white Gaussian noise $\mathbf{w}_{b_a}$ and $\mathbf{w}_{b_g}$, respectively. $\hat{\mathbf{a}}$ denotes the translational acceleration caused by the aerodynamic force acting on a flying MAV, mainly consisting of the drag force and the thrust acting on the propellers. Hence

$$\begin{aligned} \hat{\mathbf{a}} &= [a_x, a_y, a_z]^T \\ &= [k_d v_{B,x} + w_{v,x}, \quad k_d v_{B,y} + w_{v,y}, \\ &\quad \tilde{a}_z - b_{a,z} + w_{a,z}]^T \end{aligned} \quad (2)$$

where $\mathbf{w}_v$ denotes the noise of the linear drag model.

2.3 State Propagation

The estimator back-end is a simplified variant of the EKF-based back-end of the robocentric VIO [15]. It estimates the relative pose between the current body frame and the local frame of reference. The EKF state vector is defined as

$$\mathbf{x} := \begin{bmatrix} {}^{R_k} \mathbf{p}_G, & {}^{R_k} \mathbf{q}, & \mathbf{g}_{R_k}, \\ {}^{B_t} \mathbf{p}_{R_k}, & {}^{B_t} \mathbf{q}, & \mathbf{v}_{B_t}, \mathbf{b}_a, \mathbf{b}_g, k_d \end{bmatrix}. \quad (3)$$

B_t is the current body frame at time t . When a new keyframe is defined, the new reference frame R_{k+1} is set to be the same as the B_t at the time. R_k is the current reference frame. It is the k th reference frame since the estimator is initialized. G stands for the global frame. It is the first reference frame R_0 , i.e., the body frame when the first keyframe is captured after the initialization of the estimator. Note that G and W are not the same coordinate frame. They have the same origin point, but there is relative rotation between them. $\mathbf{R}_W^{(G)} \mathbf{q}$ can be calculated from $\mathbf{R}_W^{(G)} \mathbf{q} \cdot \mathbf{R}_G^{(R_k)} \mathbf{q} = [0, 0, g]^T$. ${}^{R_k} \mathbf{p}_G$ is a translation vector pointing from the origin of G to the origin of R_k , expressed in R_k . It is about the global position of R_k . ${}^{B_t} \mathbf{p}_{R_k}$ is a translation vector pointing from the origin of R_k to the origin of B_t , expressed in B_t . It is about the local position of B_t relative to R_k . ${}^{R_k} \mathbf{q}$ is the Hamilton quaternion reflecting the relative rotation between G and R_k . ${}^{B_t} \mathbf{q}$ reflects the relative rotation between R_k and B_t . $\mathbf{g}_{R_k}$ indicates the gravity vector expressed in R_k . $\mathbf{v}_{B_t}$ is the translational velocity of the IMU expressed in B_t .

Eq. (4) shows the IMU-driven state dynamics ($\dot{\mathbf{x}}$). $[\hat{\boldsymbol{\omega}}]_{\times}$ represents the skew-symmetric matrix associated with $\hat{\boldsymbol{\omega}}$. $\mathbf{w}_p$ is the process noise in position integration. $\mathbf{R}_{R_k}^{(B_t)} \mathbf{q}$ is a transformation function from ${}^{B_t} \mathbf{q}$ to $SO(3)$ rotation matrix that maps a vector expressed in B_t to its expression in R_k . $\otimes$ denotes quaternion product. $\mathbf{E}_z$ is a 3×3 diagonal matrix with $[0, 0, 1]$ as its diagonal elements. $\mathbf{E}_{x,y}$ is a 3×3 diagonal matrix with $[1, 1, 0]$ as its diagonal elements. We utilize the techniques introduced in [16] for quaternion-related calculation.

$$\begin{aligned} {}^{B_t} \dot{\mathbf{p}}_{R_k} &= -[\hat{\boldsymbol{\omega}}]_{\times} \cdot {}^{B_t} \mathbf{p}_{R_k} + \mathbf{v}_{B_t} + \mathbf{w}_p, \\ \dot{\mathbf{v}}_{B_t} &= -[\hat{\boldsymbol{\omega}}]_{\times} \cdot \mathbf{v}_{B_t} + \mathbf{R}_{R_k}^{(B_t)} \mathbf{q}^T \cdot \mathbf{g}_{R_k} \\ &\quad + \mathbf{E}_{x,y} (k_d \mathbf{v}_{B_t} + \mathbf{w}_v) + \mathbf{E}_z \hat{\mathbf{a}}, \\ {}^{B_t} \dot{\mathbf{q}}_{R_k} &= \frac{1}{2} {}^{B_t} \mathbf{q} \otimes \begin{bmatrix} 0 \\ \hat{\boldsymbol{\omega}} \end{bmatrix}, \\ \dot{\mathbf{b}}_a &= \mathbf{w}_{b_a}, \quad \dot{\mathbf{b}}_g = \mathbf{w}_{b_g}. \end{aligned} \quad (4)$$

2.4 Acceleration Measurement Update

Drag-induced acceleration in the horizontal body plane (the plane that contains the x -axis and y -axis of B) can be measured by accelerometer measurements along the x -axis and y -axis of B . Accelerometer measurements correct state propagation through the following measurement update equations of the EKF.

$$\begin{aligned} z_{a,x} &= k_d v_{B,x} + b_{a,x} + w_{a,x}, \\ z_{a,y} &= k_d v_{B,y} + b_{a,y} + w_{a,y}. \end{aligned} \quad (5)$$

The accelerometer measurement update steps along with the model-aided inertial ego-motion state propagation. The estimated horizontal states ($\mathbf{g}_{R_k}$, $v_{B,x}$, and $v_{B,y}$) do not drift over time. But they are noisy and negatively affected by the accelerometer bias $\mathbf{b}_a$. The performance of model-aided inertial ego-motion estimation is omitted in this paper. An interested reader can refer to [10]. We use the drift-free roll $\hat{\phi}$ and

pitch $\hat{\theta}$ angles to simplify the visual measurement processing, as introduced in Section 3. $\hat{\phi}$ and $\hat{\theta}$ of the current frame are calculated from $\mathbf{R}_{R_k}^{(B_t)} \mathbf{q}^T \cdot \mathbf{g}_{R_k} = \mathbf{R}(\hat{\phi}) \cdot \mathbf{R}(\hat{\theta}) \cdot [0, 0, g]^T$.

2.5 Relative Visual Measurement Update

The *Visual Processing* module outputs the visual measurements of the relative pose with respect to the keyframe, $\hat{\psi}_{key}$ and ${}^{H,key}\hat{\mathbf{t}}$, to be introduced in Section 3. $\hat{\psi}_{key}$ reflects the visual measurement of the 1-dimension rotation between the current heading frame and the keyframe heading frame. ${}^{H,key}\hat{\mathbf{t}}$ is the visual measurement of the camera translation direction expressed in the keyframe heading frame. The EKF uses these measurements in the visual measurement update through Eq. (6).

$$\begin{aligned} \mathbf{z}_r &= {}^{B_t}_{R_k} \mathbf{q} + \mathbf{w}_r, \\ \mathbf{z}_t &= \frac{\mathbf{t}_{BC} + {}^{B_t}_{R_k} \mathbf{p}_{R_k} - \mathbf{R}_{R_k}^{(B_t)} \mathbf{q}^T \cdot \mathbf{t}_{BC}}{\|\mathbf{t}_{BC} + {}^{B_t}_{R_k} \mathbf{p}_{R_k} - \mathbf{R}_{R_k}^{(B_t)} \mathbf{q}^T \cdot \mathbf{t}_{BC}\|} + \mathbf{w}_t. \end{aligned} \quad (6)$$

$\mathbf{z}_r$ is the rotation between the current body frame and the keyframe body frame. It is calculated from $\hat{\psi}_{key}$ together with the roll and pitch angles of the current frame $\hat{\phi}_{curr.}, \hat{\theta}_{curr.}$ and the keyframe $\hat{\phi}_{key}, \hat{\theta}_{key}$. $\mathbf{z}_t$ is the direction of the translational motion from the camera position at the keyframe to the current camera position, expressed in the current body frame. It is obtained by rotating ${}^{H,key}\hat{\mathbf{t}}$ to get its expression in the current body frame via $\hat{\phi}_{curr.}, \hat{\theta}_{curr.}$, and $\hat{\psi}_{key,post.}$. $\mathbf{z}_t$ is a unit vector, so the propagated camera translation is normalized accordingly. $\mathbf{t}_{BC}$ is the translation vector points from the IMU to the camera, expressed in B . In the implementation, we neglect the correlation of the elements in the three axes of the visual measurements. Thus the measurement noise matrices $\mathbf{R}_{w_t}$ and $\mathbf{R}_{w_r}$ are diagonal matrices.

The rotation update using $\mathbf{z}_r$ takes place first. And then, translation update uses $\mathbf{z}_t$. The purpose of this two-step updating is to benefit the calculation of the visual translation measurement ${}^{H,key}\hat{\mathbf{t}}$ by the more accurate *a posteriori* relative yaw angle $\hat{\psi}_{key,post.}$, as introduced in Section 3. $\hat{\psi}_{key,post.}$ is calculated from the *a posteriori* ${}^{R_k}_G \hat{\mathbf{q}}$ and $\hat{\mathbf{g}}_{R_k}$ that has been updated by $\mathbf{z}_r$.

2.6 Composition and Resetting for New Keyframe

When a new keyframe is defined, the *a posteriori* relative pose estimation ${}^{B_t}_{R_k} \hat{\mathbf{p}}_{R_k}$ and ${}^{B_t}_{R_k} \hat{\mathbf{q}}$ are composed to the global pose, as shown in Eq. (7).

$$\begin{aligned} {}^{R_{k+1}}_G \mathbf{q} &= {}^{B_t}_{R_k} \hat{\mathbf{q}} \otimes {}^{R_k}_G \mathbf{q}, \\ {}^{R_{k+1}}_G \mathbf{p}_G &= \mathbf{R}_{R_k}^{(B_t)} \hat{\mathbf{q}}^T \cdot {}^{R_k}_G \mathbf{p}_G + {}^{B_t}_{R_k} \hat{\mathbf{p}}_{R_k} \end{aligned} \quad (7)$$

The current body frame B_t becomes the new reference frame R_{k+1} . The expression of the gravity vector $\mathbf{g}_{R_{k+1}}$ in R_{k+1} is calculated as $\mathbf{g}_{R_{k+1}} = \mathbf{R}_{R_k}^{(B_t)} \hat{\mathbf{q}}^T \cdot \hat{\mathbf{g}}_{R_k} \cdot {}^{B_t}_{R_{k+1}} \mathbf{p}_{R_{k+1}}$ and ${}^{B_t}_{R_{k+1}} \mathbf{q}$ are set to a zero vector and a unit quaternion whose vector part is a zero vector, respectively. Their corresponding elements in the covariance matrix are set to zeros too.

3 VISUAL RELATIVE POSE ESTIMATION

In this section, we describe the proposed method for estimating the yaw angle and the direction of translation relative to the keyframe. The relative yaw and translation are calculated separately, as shown in Fig. 1.

3.1 Keyframe-based Feature Tracking

We follow the same idea of keyframe-based feature detection and tracking strategy as [5, 12]. Kanade-Lucas-Tomasi (KLT) [17] is utilized as the feature tracker to continuously track FAST features [18] between frames. If the number of points tracked in the current frame falls below a threshold, the current frame is defined as the new keyframe. A fixed number of uniformly distributed FAST points are detected in a new keyframe by evenly splitting the image into several regions while keeping the same number of good features in each region. These newly detected points are then added to a database for tracking in the coming frames.

3.2 Linear Relative Yaw Calculation

Based on the epipolar geometry of a pair of corresponding points, the linear 8-point algorithm [19] calculates the essential matrix which helps extracting the 3D relative rotation and the 2D direction of translation. If two relative orientation angles are known, the remaining angle and the direction of translation can be calculated linearly with 5-point pairs [20]. Slightly different from [12], our linear 5-point algorithm projects point coordinates on a unit sphere in the heading frame H .

The pixel location of a feature point is first undistorted and normalized using the distortion parameters and the intrinsic matrix of the camera. We then obtain the homogeneous coordinates of the feature points expressed in the camera frame, ${}^C \tilde{\mathbf{p}}_{key}$ and ${}^C \tilde{\mathbf{p}}_{curr.}$. The epipolar constraint is given by

$${}^C \tilde{\mathbf{p}}_{curr.}^T [{}^{C,curr.} \mathbf{t}_\times] \mathbf{R}_{C,key}^{C,curr.} {}^C \tilde{\mathbf{p}}_{key} = 0 \quad (8)$$

where $[{}^{C,curr.} \mathbf{t}_\times]$ is the skew-symmetric matrix of the translation vector ${}^{C,curr.} \mathbf{t}_{key}$ that points from the origin of the current camera frame to the origin of the keyframe camera frame, expressed in current camera frame.

Using the camera extrinsic rotation matrix $\mathbf{R}_C^B$ and the estimated roll $\hat{\phi}$ and pitch $\hat{\theta}$ angles of the keyframe and the current frame, we can express the feature point coordinates in the heading frame H , as

$$\begin{aligned} {}^H \hat{\mathbf{p}}_{key} &= \mathbf{R}_{\hat{\theta},key}^T \mathbf{R}_{\hat{\phi},key}^T \mathbf{R}_C^B {}^C \tilde{\mathbf{p}}_{key}, \\ {}^H \hat{\mathbf{p}}_{curr.} &= \mathbf{R}_{\hat{\theta},curr.}^T \mathbf{R}_{\hat{\phi},curr.}^T \mathbf{R}_C^B {}^C \tilde{\mathbf{p}}_{curr.} \end{aligned} \quad (9)$$

${}^H \hat{\mathbf{p}}_{key}$ and ${}^H \hat{\mathbf{p}}_{curr.}$ are normalized to unit vectors ${}^H \bar{\mathbf{p}}_{key}$ and ${}^H \bar{\mathbf{p}}_{curr.}$, which are equivalent to the 3D coordinates of the feature points projected onto the unit sphere. They describe the directions of the feature points in the heading frame H ,

so we refer to them as *point vectors*. The epipolar constraint equation in H is given by

$${}^H\bar{\mathbf{p}}_{curr.}^T \lfloor {}^{H,curr.}\mathbf{t}_\times \rfloor \mathbf{R}_{H,key}^{H,curr.} {}^H\bar{\mathbf{p}}_{key} = 0 \quad (10)$$

where $\mathbf{R}_{H,key}^{H,curr.}$ is the relative rotation between two heading frames, which equals to $\mathbf{R}_{\psi_{key}}$. In Eq. (10), the essential matrix $\mathbf{E}_H = \lfloor {}^{H,curr.}\mathbf{t}_\times \rfloor \mathbf{R}_{\psi_{key}}$ has six entries defined up-to-scale, which can be linearly solved with a minimum of five points [20]. The visual measurement of relative yaw $\hat{\psi}_{key}$ is then calculated from $\mathbf{E}_H$.

Both [5] and [12] reject outliers in the tracked points using the prior relative pose based on state propagation. They apply a threshold on the 2D distance from each tracked point to the epipolar line or the residuals of the epipolar constraint equation. This scheme is faster than iterative random sample consensus (RANSAC) outlier rejection. However, aggressive maneuvers may cause fewer feature points to be detected and correctly tracked. Due to the biases of the IMU measurements and the simplified drag model, the propagated relative translational motion would diverge if without enough vision updates. Consequently, the prior relative pose is not accurate enough for outlier rejection. Therefore, we employ RANSAC in the linear 5-point algorithm to calculate the relative yaw angle. Setting the number of random trials to a reasonably small number limits the required computation cost.

3.3 Linear Relative Translation Direction Calculation

During implementing the 5-point algorithm and testing it on EuRoC, it was observed that the translation direction vectors calculated together with relative yaw were noisy. The standard deviation of the direction error can be more than 30 degrees compared to the ground-truth motion direction. Instead of using this translational direction in the EKF update, we take the relative yaw angle as a known value and re-calculate the translation vector ${}^{H,key}\hat{\mathbf{t}}$. As introduced before, the visual translation measurement update happens after the rotation update. So we can calculate ${}^{H,key}\hat{\mathbf{t}}$ after the rotation update and make use of the updated *a posteriori* relative yaw angle $\hat{\psi}_{key,post.}$. This is achieved by rotating ${}^H\bar{\mathbf{p}}_{curr.}$ from the current heading frame to keyframe heading frame by $\mathbf{R}(\hat{\psi}_{key,post.})$. Then we get the epipolar constraint equation in the keyframe heading frame as

$${}^{H,key}\bar{\mathbf{p}}_{curr.}^T \lfloor {}^{H,key}\mathbf{t}_\times \rfloor {}^{H,key}\bar{\mathbf{p}}_{key} = 0. \quad (11)$$

There are only three up-to-scale entries in the translation-only essential matrix $\mathbf{E}_{H,key} = \lfloor {}^{H,key}\mathbf{t}_\times \rfloor$ in Eq. (11). Therefore, ${}^{H,key}\hat{\mathbf{t}}$ can be solved by a minimum of two point correspondences, as in [21].

We solve ${}^{H,key}\hat{\mathbf{t}}$ from the inlier point pairs of the 5-point RANSAC. In order to calculate translation, corresponding point vector pairs with big enough angles are necessary. After rotating the inlier point vectors into the keyframe heading frame, the point vector pairs are checked. Only pairs with

angles beyond a certain threshold are used. In our implementation, the threshold is 5 degrees. Note that Eq. (11) has two mirrored solutions. The true direction can be determined by triangulating features and choosing the solution with all the features in front of the camera. To be more efficient, we use the propagated ${}^{B_t}\mathbf{p}_{R_k}$ to determine the direction of ${}^{H,key}\hat{\mathbf{t}}$. The solution that has a smaller angle with the expression of ${}^{B_t}\mathbf{p}_{R_k}$ in the heading frame is selected.

4 EXPERIMENTAL RESULTS

We evaluate the proposed ego-motion estimator on the widely recognized EuRoC MAV dataset. This dataset is comprised of multiple sequences with varying lighting conditions, and a maximum flight speed of 2.3 m/s. The proposed approach uses synchronized left-camera images and IMU measurements.

4.1 Data Pre-processing

In the EuRoC dataset, the IMU does not coincide with the forward-right-down body frame definition. As a result, the accelerometer's x -axis and y -axis do not directly measure the drag force in the body horizontal plane. A body-IMU rotation matrix is needed to transfer the accelerometer's measurements to the body frame. We assume that the coordinate frame of the reflective markers mounted on the MAV approximately coincides with the body frame. Because the origin of our body frame is the same as the IMU frame, we use the sensor rotational extrinsic data to rotate IMU measurements and ground truth to the body frame.

The proposed estimator is initialized just before taking off to comply with the drag force model, which only stands when the MAV is in flight. Visual updating starts after the second keyframe is captured. The initial value of k_d was set to -0.2 for all the sequences. This value is close to the least-square solution calculated by the ground-truth velocity of the *Machine Hall 05 (MH05)* sequence.

4.2 Results and Discussion

The main evaluation results of the proposed estimator in terms of time efficiency and accuracy are listed in Table 1. The C++ code of this work is implemented based on the open-sourced code of [22]. The proposed estimator is compared with a state-of-the-art (SOTA) VIO MSCKF [2] implemented by the same open-sourced repository [22]. The top group of Table 1 (from ① to ③) shows the test results of the proposed estimator with three different settings in the vision front-end. The bottom group (④ and ⑤)) shows the outputs of MSCKF in two settings. Here we only compare with MSCKF since the gap in accuracy is big. The comparison of MSCKF with other VIO solutions can be found in [9].

The time consumption measurements shown in the second and the third columns of Table 1 were collected during the tests on *MH05* sequence of the EuRoC dataset, running on a laptop computer. The root-mean-square error (RMSE) of absolute translation error (ATE) is the metric of estimation

Table 1: Accuracy and time efficiency of the proposed method (top group, rows 1 to 3) and the baseline method (bottom group, rows 4 and 5). *V101* to *MH05* shown in the eleven columns from the right are the names of flight sequences of the EuRoC MAV dataset. The data below the sequence names show the root-mean-square errors (RMSE) of the absolute translation errors (ATEs) of the estimated trajectories. **Bold** represents the overall best and underline represents the best of the proposed method.

ID	ATT ¹	AVT ²	NFP ³	V101	V102	V103	V201	V202	V203	MH01	MH02	MH03	MH04	MH05
①	4.41	3.11	30	5.12	1.19	8.04	4.21	2.11	5.90	4.70	13.4	12.0	2.37	2.76
②	7.25	5.81	30	<u>1.26</u>	<u>1.17</u>	<u>3.11</u>	<u>2.38</u>	<u>1.48</u>	<u>2.91</u>	<u>3.60</u>	13.2	11.2	<u>2.12</u>	1.56
③	12.9	11.5	250	2.04	1.76	5.37	4.56	2.80	7.76	13.5	<u>1.67</u>	10.4	2.37	<u>1.35</u>
④	7.93	3.82	51	0.16	0.13	0.12	245	0.13	0.16	38.0	5.20	130	2.35	1.00
⑤	17.4	7.58	199	0.09	0.09	0.11	0.12	0.10	0.20	0.34	0.24	58.5	0.65	1.54

¹ Average total time (ATT) consumption (millisecond) of processing after receiving a new frame.

² Average vision-related time (AVT) consumption (millisecond) in processing a single frame, including feature tracking, feature detection, and outlier rejection of feature points.

³ Number of feature points (NFP). For the proposed approach (top group), the shown number is the number of feature points to detect in a new keyframe. For MSCKF (bottom group), the shown number is the average number of tracked points in each frame.

accuracy. The RMSEs of ATEs of the eleven flight sequences of the EuRoC dataset are shown in Table 1. The alignment of the estimated trajectory and the ground-truth trajectory has 4 degrees of freedom (yaw and 3-d translation). The calculation is conducted by [23].

The upper group of Table 1 (from ① to ③) shows the test results of the proposed estimator with three different settings in the vision front-end. For ① and ②, 30 feature points are detected in a keyframe, and a new keyframe is defined when the number of inlier points is below 10. For ③, 250 feature points are detected in a keyframe, and a new keyframe is defined when there are fewer than 60 inliers. ① corresponds to the estimator that uses images that are downsampled to half of the original resolution. And it does not have histogram equalization as image preprocessing. The rest settings (② and ③) and MSCKF (④ and ⑤) use images with original resolution and preprocessed by histogram equalization. Thus the vision processing time consumption of ① is much lower than ② while sacrificing the accuracy. It is an extreme case that minimizes visual processing. In the 5-point RANSAC of ①, ②, and ③, six point pairs are selected as a sample. The total number of random trials is set to six. If the highest inlier rate is more than 60%, this sample's inlier point pairs are used to calculate the essential matrix.

Comparing ② and ③, it is obvious that more feature points do not necessarily improve the accuracy of the proposed estimator. This is different from MSCKF. As shown by ④ and ⑤, ⑤ detects and tracks more points and has higher accuracy. The reason is that the vision update of MSCKF is triggered when a point loses tracking or leaves the field of view. Thus a bigger number of points means more times of updating. But for the proposed estimator, vision update happens for every frame and the visual measurement is calculated from all points tracked in a frame. In this case, fewer points do not necessarily lead to a less accurate essential matrix. On

the contrary, a big number of tracked points with tracking noise would deteriorate the accuracy. So the proposed estimator is more suitable for applications where the computational power for visual processing is very limited and only a small number of points can be detected and tracked.

For ②, the estimated trajectories of *MH02* and *MH03* are less accurate than other sequences. The trajectory errors are mainly caused by the big estimation errors at the beginning of the sequences. For *MH02*, the ATE of the second half of the estimated trajectory is 3.243. For *MH03*, after deleting 20% of the estimated trajectory from the beginning of the sequence, the ATE of the rest part of the trajectory is 2.979. From Fig. 4, we can see that the estimation converges to be accurate after around 30 seconds since taking off. So in real-world applications, it is better to have safety measures for the possible big drifts at the beginning of a flight, for example, a space without obstacles close by.

From Table 1, we notice that, in the sequences *V201*, *MH01*, and *MH03*, MSCKF ④ drifts after very slow motion, which leads to big trajectory errors. But in general, the proposed approach is less accurate than MSCKF. Despite that, as shown in Fig. 3, Fig. 4, and Fig. 5, the accuracy is acceptable for short-term ego-motion estimation. The advantages of the proposed estimator are time efficiency and robustness. As shown in Table 1, ① and ② consume less time than MSCKF ④. Another fact worth mentioning is that we observed in experiments that our 5-point RANSAC implementation runs slower than the OpenCV 8-point RANSAC adopted by MSCKF. It can be attributed to the highly optimized OpenCV library function. There can be a space to further shorten the time consumption by optimizing our implementation. As for robustness, the proposed estimator does not have big drifts as ④ when testing on the sequences *V201*, *MH01*, and *MH03*. As the proposed estimator does not map the environment, sudden loss tracking of many or all points

does not require re-initializing the system. In addition, the proposed estimator maintains accuracy even if there is not visual measurement anymore. As shown in Fig. 5, we cut off the camera image stream at the 52nd second after taking off, and the accuracy basically maintains. Obvious estimation errors are only observed in the velocity estimation of the z -axis in the last ~ 12 seconds of flight.

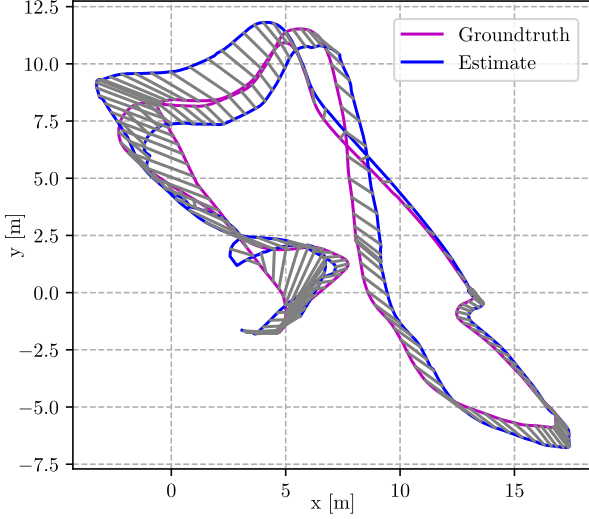


Figure 3: Estimated trajectory of *MH05* sequence by the proposed estimator ② in Table 1.

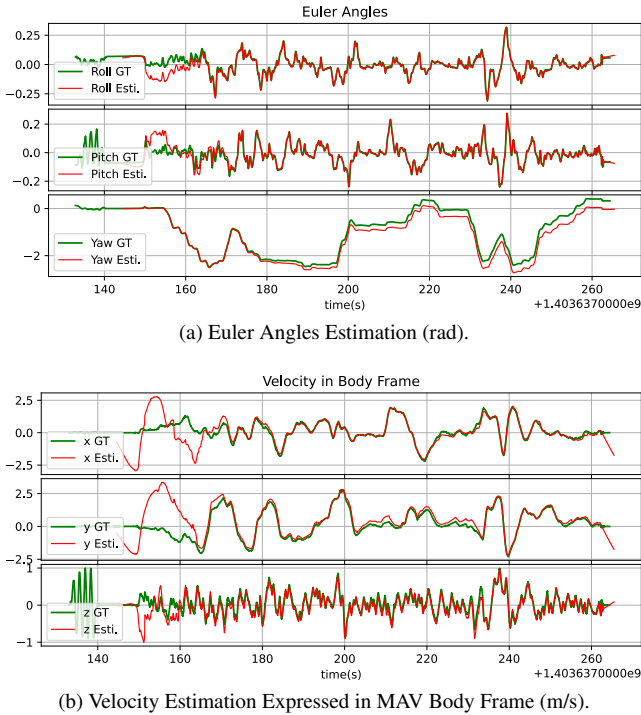


Figure 4: Estimated attitude and velocity of *MH03* sequence by the proposed estimator ① in Table 1.

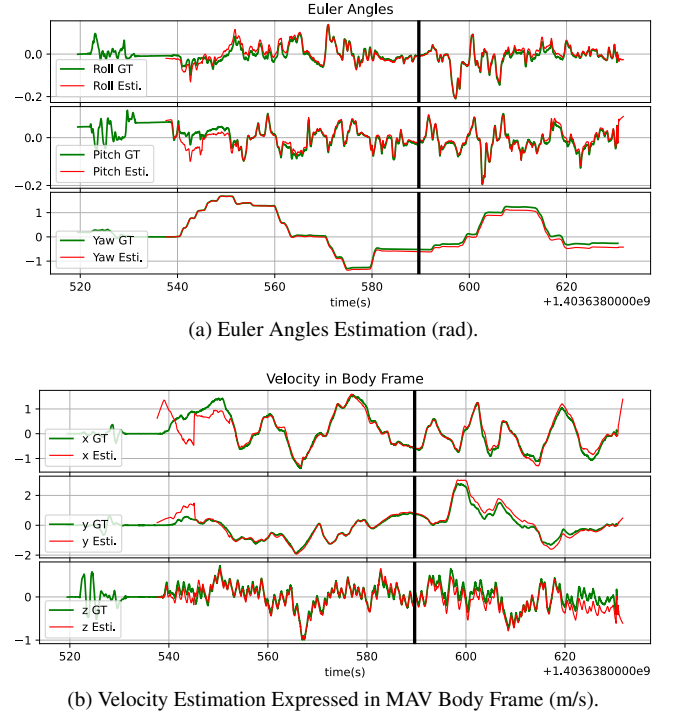


Figure 5: Estimated attitude and velocity of *MH05* sequence by the proposed estimator ② in Table 1. Camera images are no longer used by the estimator after ~ 590 seconds.

5 CONCLUSION AND FUTURE WORK

In this paper, we present a model-aided ego-motion estimator that is friendly to multirotor MAVs with limited processing power onboard. Minimized visual processing and a simple EKF-based backend that performs loosely-coupled visual-inertial fusion lead to high time efficiency. The proposed estimator is less accurate than a SOTA approach in comparison, but it can be enough for short-term navigation. Autonomous drone racing is a potential application. The proposed estimator's small requirement for processing power benefits other algorithms running onboard. It has acceptable drifts in estimating flight trajectories, which can be corrected by the racing gates that can serve as stationary landmarks. The estimation accuracy rarely drops in short term when the vision information becomes absent, implying that the estimator can stay functional when there is no valid feature point due to the motion blur in agile maneuvers.

REFERENCES

- [1] Shuo Li, Erik van der Horst, Philipp Duernay, Christophe De Wagter, and Guido C H E de Croon. Visual model-predictive localization for computationally efficient autonomous racing of a 72-gram drone. 2019. arXiv: 1905.10110.
- [2] Mingyang Li and Anastasios I Mourikis. High-precision, consistent EKF-based visual-inertial odom-

- etry. *The International Journal of Robotics Research*, 32(6):690–711, 2013.
- [3] Tong Qin, Peiliang Li, and Shaojie Shen. VINS-Mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.
- [4] 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(10):1053–1072, 2017.
- [5] Dinuka Abeywardena, Shoudong Huang, Ben Barnes, Gamini Dissanayake, and Sarath Kodagoda. Fast, on-board, model-aided visual-inertial odometry system for quadrotor micro aerial vehicles. *IEEE International Conference on Robotics and Automation*, pages 1530–1537, 2016.
- [6] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *IEEE International Conference on Robotics and Automation*, pages 15–22, 2014.
- [7] Stephan Weiss and Roland Siegwart. Real-time metric state estimation for modular vision-inertial systems. In *IEEE International Conference on Robotics and Automation*, pages 4531–4537, 2011.
- [8] Matthias Faessler, Flavio Fontana, Christian Forster, Elias Mueggler, Matia Pizzoli, and Davide 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.
- [9] Jeffrey Delmerico and Davide Scaramuzza. A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. *IEEE International Conference on Robotics and Automation*, pages 2502–2509, 2018.
- [10] Dinuka Abeywardena, Sarath Kodagoda, Gamini Dissanayake, and Rohan Munasinghe. Improved state estimation in quadrotor mavs: A novel drift-free velocity estimator. *IEEE Robotics & Automation Magazine*, 20(4):32–39, 2013.
- [11] Matthias Faessler, Antonio Franchi, and Davide Scaramuzza. Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *IEEE Robotics and Automation Letters*, 3(2):620–626, 2017.
- [12] James Svacha, Giuseppe Loianno, and Vijay Kumar. Inertial yaw-independent velocity and attitude estimation for high-speed quadrotor flight. *IEEE Robotics and Automation Letters*, 4(2):1109–1116, 2019.
- [13] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016.
- [14] Marc Rigter, Benjamin Morrell, Robert G Reid, Gene B Merewether, Theodore Tzanetos, Vinay Rajur, KC Wong, and Larry H Matthies. An autonomous quadrotor system for robust high-speed flight through cluttered environments without GPS. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5227–5234, 2019.
- [15] Zheng Huai and Guoquan Huang. Robocentric visual-inertial odometry. *The International Journal of Robotics Research*, 41(7):667–689, 2022.
- [16] Joan Sola. Quaternion kinematics for the error-state kalman filter. *arXiv preprint arXiv:1711.02508*, 2017.
- [17] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. *International Journal of Computer Vision Technical Report*, 1991.
- [18] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, pages 430–443, 2006.
- [19] H Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, 1981.
- [20] Friedrich Fraundorfer, Petri Tanskanen, and Marc Pollefeys. A minimal case solution to the calibrated relative pose problem for the case of two known orientation angles. In *European Conference on Computer Vision*, pages 269–282. Springer, 2010.
- [21] Chiara Troiani, Agostino Martinelli, Christian Laugier, and Davide Scaramuzza. 2-Point-based outlier rejection for camera-IMU systems with applications to micro aerial vehicles. In *IEEE International Conference on Robotics and Automation*, pages 5530–5536, 2014.
- [22] Patrick Geneva, Kevin Eickenhoff, Woosik Lee, Yulin Yang, and Guoquan Huang. OpenVINS: A research platform for visual-inertial estimation. In *IEEE International Conference on Robotics and Automation*, 2020.
- [23] Zichao Zhang and Davide Scaramuzza. A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7244–7251. IEEE, 2018.

Analysis of admittance control for a fully-actuated, tactile UAV

Iris Liu*, Suvarna Swaraj, Karl Stol and Salim Al-zubaidi

Department of Mechanical and Mechatronics Engineering, University of Auckland, New Zealand

ABSTRACT

Abstract— The main contribution of this paper is the tuning of an admittance controller on a compact, fully-actuated, multirotor Unmanned Aerial Vehicle (UAV) for the purpose of achieving optimal, intuitive, physical human drone interaction (PHDI). The controller permits a human to physically translate and rotate the UAV in flight while minimizing reaction forces and moments. Being a fully-actuated UAV allows the translation and rotation to be decoupled. The performance of the admittance controller was validated through simulations and experimental testing; optimal reaction forces of 0.7% and 1.3% of the UAV's weight were achieved for translation in the horizontal axis in simulations and free flight tests, respectively.

1 INTRODUCTION

The prevalence of physical human drone interaction (PHDI) has been rising in recent years due to their importance in various applications such as search and rescue [1], augmented drone ball sports [2], assisting the visually impaired [3], and enhancing the virtual reality (VR) gaming experience through haptic feedback [4]. Literature focused on physical or touch based interaction with UAVs improves knowledge of different techniques used to estimate the external forces and torques applied to the UAV during an interaction, and controllers which regulate the UAV's resultant behavior.

The performance of PHDI can be influenced by the UAV rotor configuration. Underactuated UAVs are categorized as having less actuators than the number of degrees of freedom (DoF). They are widely accessible and simply constructed, however, have coupled rotational and translational dynamics. The intrinsic underactuation of such conventional quadrotors may inhibit the PHDI performance and cause safety concerns [5]. Existing studies have explored PHDI with underactuated UAVs such as use of small quadcopters that act as haptic-input devices for greater immersion of the user in a VR game [4]. Similarly, Braley et al. [6] investigated stackable nanocopters which can be physically deformed by the user for the purpose of representing voxel grids for 3D spatial transformations. Underactuated UAVs are suitable for PHDI when they are required to be small, compact, inexpensive, and for



Figure 1: User physically interacting with a multirotor UAV

a swarm. However, interactions that lack intuition can result due to their coupled translational and rotational dynamics, such as for a safe-to-touch quadcopter built by Abtahi et al. [7].

An opportunity exists to exploit the capabilities of fully-actuated UAVs for PHDI, where the translational and rotational dynamics are decoupled. Improved omni-directional motion and agility are key advantages of fully-actuated UAVs over their underactuated counterparts [8]. In a wider context, research into physical drone interactions have been conducted on a fully-actuated drone [9], and the ability to independently control the linear and angular accelerations allowed for precise interactions with static objects to be made.

Admittance control has been widely used in human robot interaction applications for generating compliant trajectories such as stabilized stair climbing for humanoid robots [10] and control of a robotic exoskeleton [11]. Augugliaro et al. [12] demonstrated the ability of an admittance controller to adjust a quadcopter's reference trajectory to comply with an interaction. This indicates the merit of extending research into using admittance control to regulate the behavior of a fully-actuated UAV during interaction. An admittance controller seeks to make the closed loop behavior of the UAV match a mass-spring-damper system using the estimated external wrench during the interaction. The admittance controller parameters, mass, spring, and damping, can be tuned intuitively based upon the behavior desired from the UAV, as shown in [13]. Rajappa et al. [5] simulated use of admittance control to regulate the behavior of a fully-actuated hexarotor. To the best of the authors' knowledge, this is the only study in literature currently that explores the use of admittance control for a fully-actuated UAV in the area of PHDI. The work in this paper differs as the concept was experimentally tested and optimized. The hexarotor in [5] was able to translate to a new position with minimal changes to its attitude; hence

*Email address: sliu142@aucklanduni.ac.nz

showcasing the dexterity of a fully-actuated UAV in PHDI. A force sensor ring was installed on the UAV to perform external wrench estimation. Due to the potential loss in flight time, added weight, and power consumed, additional hardware is not a practical option for interaction wrench detection in this research. Software based methods of interaction detection to estimate the external wrench could, therefore, be more appropriate. These include an approach proposed by Tomic et al. [14] where a hybrid momentum-acceleration-based estimator was used by exploiting onboard sensors.

The work conducted in this paper involves the tuning of an admittance controller for regulating the behavior of a fully-actuated, stacked, fixed-tilt, octorotor during physical interaction. The performance of the controller is validated in simulation and experiments for multiple translational and rotational axes. Additionally, simulations and optimization in the roll-axis showcase the unique capability of a fully-actuated UAV to station keep at a non-zero roll angle after interaction, allowing its use in a greater range of applications.

2 MULTIROTOR UAV MODEL

The fully-actuated multirotor used in this study is based on the octorotor in [15], featuring a compact, stacked rotor design (Figure 2). It has a flying mass of 985g and hover time of 14 minutes. To achieve safe and effective PHDI, an interaction frame and propeller guards were designed and attached onto the UAV, pictured in Figure 3. The added frame results in an additional 280g in the UAV's mass; a total flight mass of 1265g. The safety and mechanical design requirements of the interaction frame achieved are: a minimum clearance of 20 mm between the added features and the rotor blades, and a minimum clearance of 200 mm between the handle and the rotor blades. The position vector is defined as $\xi = [x, y, z]$ and the rotational vector is expressed using the Euler angle conventions of $\eta = [\phi, \theta, \psi]$, corresponding to roll, pitch, and yaw, respectively.

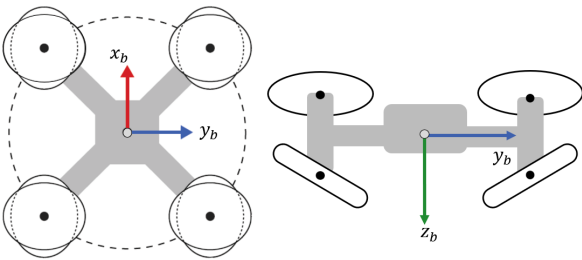


Figure 2: Schematic of stacked multirotor and configuration of tilted rotors about the orthogonal axes

For the purpose of the work conducted in this paper, all external forces measured are assumed to be from the human interaction and disturbances, such as wind, are negligible.

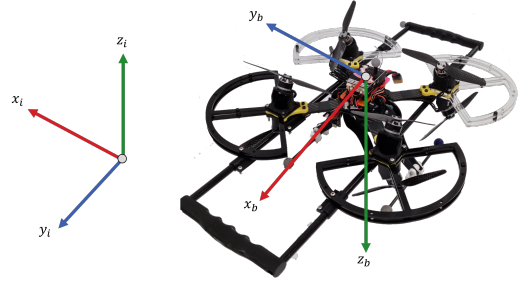


Figure 3: PHDI setup with a fully-actuated stacked multirotor and the designed interaction frame based on the concept from [15]. Subscripts b and i denote body and inertial frames, respectively.

3 CONTROL SYSTEM ARCHITECTURE

The control system architecture design, as shown in Figure 4, consists of five main components:

1. The standard cascaded P-PID controllers of the PX4 firmware, namely position and attitude controllers, which provide position/velocity and angle/angular rate control, respectively.
2. The motor mixer which produces the motor setpoints in the form of PWM signals.
3. Human external wrench from the physical interaction (f_{ext} and τ_{ext}).
4. An external wrench estimator, based on the one used by Tomić et al. [14], which takes in the measured acceleration ($\ddot{\xi}_{meas}$) and angular velocity ($\dot{\eta}_{meas}$) to estimate the forces and moments, f_e and τ_e , respectively, applied by the human during the interaction. The estimator utilizes a hybrid momentum-acceleration-based approach, together with the onboard inertial measurement unit (IMU) and the UAV's dynamics model to provide wrench estimation.
5. The admittance controller, which takes in the estimated wrench and calculates new position and attitude setpoints, $\xi_{SP,AC}$ and $\eta_{SP,AC}$, which provide references for the subsequent controllers.

3.1 Admittance Controller

Admittance controllers are commonly used in human-robot interactions to achieve compliant trajectories based on the applied interaction forces and moments. A second order mass-spring-damper system is used to describe the desired characteristics of the closed-loop position and attitude control systems.

Equation 1 represents the desired closed-loop system dynamics:

$$M \begin{bmatrix} \ddot{\xi} \\ \ddot{\eta} \end{bmatrix} + C \begin{bmatrix} \dot{\xi} \\ \dot{\eta} \end{bmatrix} + K \begin{bmatrix} \xi \\ \eta \end{bmatrix} = \begin{bmatrix} f_e \\ \tau_e \end{bmatrix} \quad (1)$$

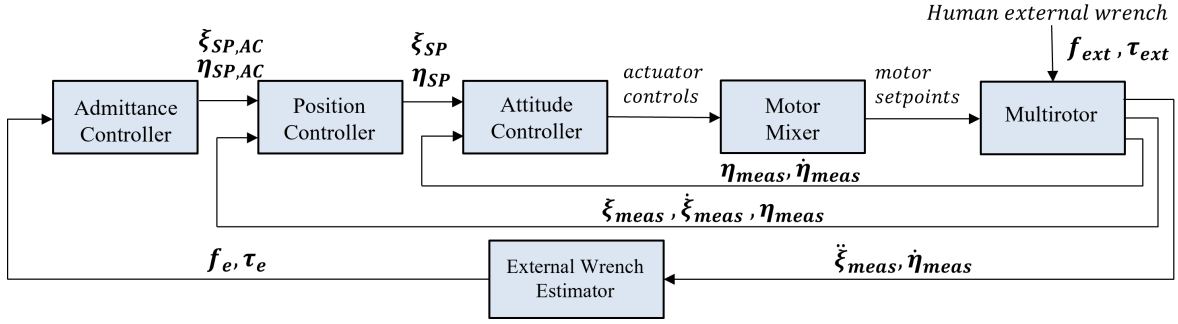


Figure 4: Control system architecture

where

$$\begin{aligned} M &= \text{diag} \{m_x \ m_y \ m_z \ m_\phi \ m_\theta \ m_\psi\}, \\ C &= \text{diag} \{c_x \ c_y \ c_z \ c_\phi \ c_\theta \ c_\psi\}, \\ K &= \text{diag} \{k_x \ k_y \ k_z \ k_\phi \ k_\theta \ k_\psi\}, \\ f_e &= \text{diag} \{f_{e,x} \ f_{e,y} \ f_{e,z} \ f_{e,\phi} \ f_{e,\theta} \ f_{e,\psi}\}, \\ \tau_e &= \text{diag} \{\tau_{e,x} \ \tau_{e,y} \ \tau_{e,z} \ \tau_{e,\phi} \ \tau_{e,\theta} \ \tau_{e,\psi}\}. \end{aligned}$$

M, C, K, f_e, τ_e correspond to the mass, damping, spring, and estimated force and moment matrices, respectively. Note that the parameters in the y and pitch axes have been set as equal to the corresponding x and roll axes, respectively, as similar behavior is expected. As the objective is to reduce the experienced reaction wrench from the UAV, K has been set to the zero matrix to enable passive accommodation control; this has been seen in similar studies involving admittance control for human UAV interactions [16].

The output of the admittance controller is a vector of set-points, therefore the control laws for the position and attitude are Equations 2 and 3, respectively.

$$\xi_{SP,AC}(s) = \frac{f_e(s)}{Ms^2 + Cs} \quad (2)$$

$$\eta_{SP,AC}(s) = \frac{\tau_e(s)}{Ms^2 + Cs} \quad (3)$$

4 SIMULATION

4.1 Simulation Model

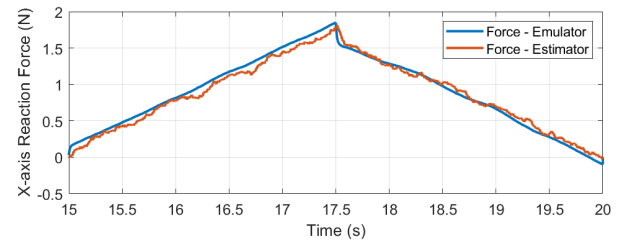
From Figure 4, the simulated multirotor models the frame and each rotor as independent rigid bodies. Aerodynamic forces are limited to quasi-steady thrusts and torques acting at each rotor; aerodynamic forces due to UAV motion through still air are ignored. The human external wrench is modeled using a human external wrench emulator. For the force emulator, this takes an input of the UAV's final position, ξ_f , and duration of interaction, t_d , as specified by the user. The reference acceleration profile, $\ddot{\xi}_{ref}$, is computed as shown in Equation 4. As shown in Equation 5, a PID controller, $C(s)$, acts on $E_\xi(s)$, the difference between the reference, ξ_{ref} , and the actual position, ξ_{meas} , and outputs the simulated external forces, f_{ext} , to be applied to the multirotor plant. The

moment emulator follows the same method to simulate external moments, τ_{ext} .

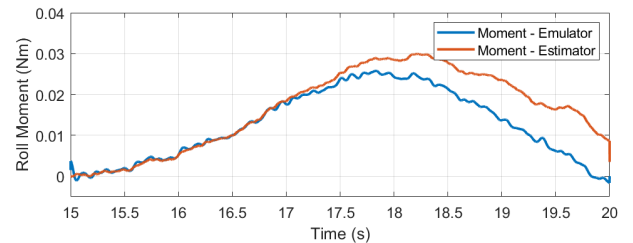
$$\ddot{\xi}_{ref}(s) = \frac{4\xi_f}{t_d^2} \quad (4)$$

$$f_{ext} = C(s)E_\xi(s) \quad (5)$$

In order to achieve PHDI, the external wrench must be estimated accurately. The estimated forces and moments are compared against the output of the human emulator. Figures 5(a) and (b) illustrate the wrench estimator and human emulator comparison for the x and roll axes, respectively. In both cases, the estimated forces and moments appear to follow the emulated profiles, thus, the estimator appears to accurately estimate the forces and moments. A deviation is observed in the second half of moment estimation in Figure 5(b). This could be attributed to tuning of gains required for the magnitude of the moments to match.



(a)



(b)

Figure 5: External wrench estimation performance for (a) x -axis and (b) roll-axis



Figure 6: Parameter sweep of m_x and c_x values for (a) a 1m x -axis drag and m_ϕ and c_ϕ for (b) a 5deg roll-axis drag simulations to optimize for minimal RMS reaction force and moment, respectively, during interaction. Red circle indicates the optimal point for each of the parameter sweeps.

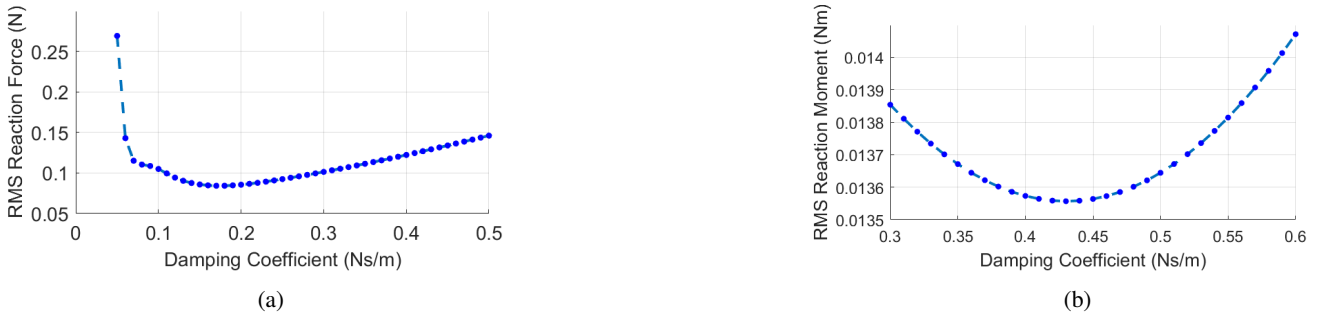


Figure 7: Relationship between changing c_x with RMS reaction force for (a) a 1m x -axis drag and c_ϕ with RMS reaction moment for (b) a 5deg roll-axis drag simulation at optimal m_x and m_ϕ of zero

4.2 Test Procedure

The work in this paper focuses on a drag interaction which involves specifying a final displacement, ξ_f and η_f , for the UAV to reach at the end of the interaction, and begins and ends at rest. The duration of the interaction was set to $t_d = 5s$ and began at time, $t = 15s$. Note that for the purpose of understanding, positive z has been defined as upwards, as defined in the inertial coordinate system.

Drag interactions were simulated for five single-DoF cases:

1. x -translational drag - $\xi_f = (1, 0, 0)m$
2. z -translational drag up - $\xi_f = (0, 0, 1)m$
3. z -translational drag down - $\xi_f = (0, 0, -1)m$
4. Yaw-axis drag - $\eta_f = (0, 0, 45)deg$
5. Roll-axis drag - $\eta_f = (5, 0, 0)deg$

The duration of interaction for analysis of data is defined as when the admittance controller first updates new setpoints until it stops. To achieve a high level of compliance, the aim is to obtain as low a reaction force or moment as possible, thus the objective function is to minimize the estimated force or moment during interaction. The scope of the paper focuses on the interaction stage, over which the root mean square (RMS) force and moment are calculated.

4.3 Results and Discussion

From Figure 6, three main trends are observed in both cases:

1. For a constant mass, the RMS reaction force and moment increases with increasing damping. This aligns well with Equation 1, where the damping force is proportional to the damping coefficient for a set velocity. As the velocity in the concerned axis is fairly consistent across the tested cases, a high reaction force and moment is expected as damping increases.
2. For constant damping, the RMS reaction force and moment increases with increasing mass. A heavier mass suggests a larger force and moment is required to accelerate the system, hence resulting in this observed trend.
3. Turning points are observed at low damping values where the RMS reaction force and moment increases significantly. A finer c_i step size of $0.01Ns/m$ was then used to capture the trends near the optimal point. Figures 7(a) and (b) illustrate the local minima by sweeping through damping coefficients. A clear turning point is observed for both cases and the damp-

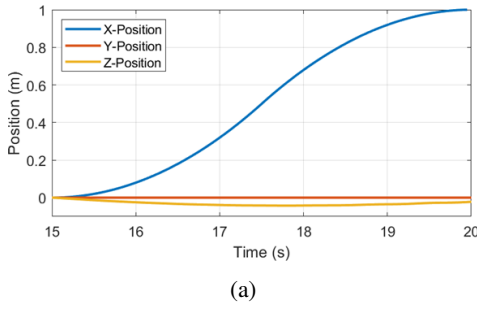
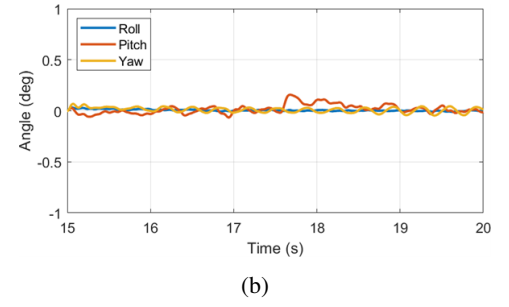
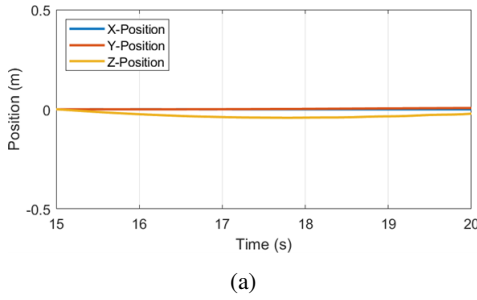
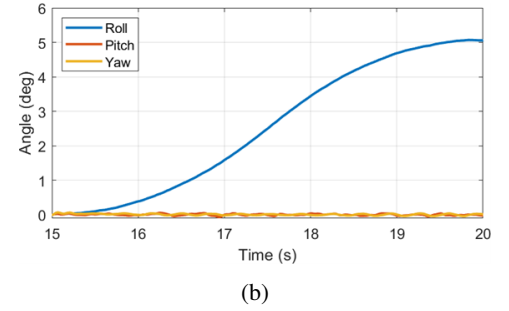
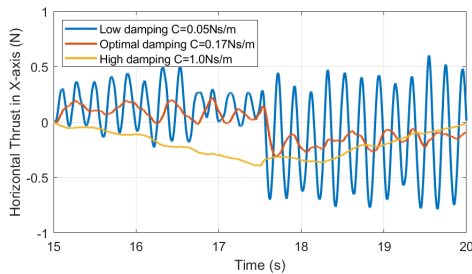
Figure 8: (a) Position and (b) attitude plots for the 1m x -axis drag

Figure 9: (a) Position and (b) attitude plots for the 5deg roll-axis drag

ing coefficient that produced the lowest RMS reaction force/moment is considered the optimal. Through observation of position and attitude plots, a large amount of oscillations was noted for low damping cases. This could be due to amplification of uncertainties and the dynamics which in turn cause deviations between the setpoint and the measured position and attitude, resulting in larger control efforts to stabilize the system. This is further shown by the control input, horizontal thrust, as shown in Figure 10. At low damping, $c_x = 0.05 \text{ N s/m}$, severe oscillations depict marginally stable behavior. Hence, the high peaks at low damping coefficients suggests that a compromise is required between the level of oscillations that can be tolerated and a comfortable reaction force.

Figure 10: Control input of horizontal thrust for three levels of damping for a 1m x -axis drag. Calculated by taking the x -axis horizontal component of the total thrust

Figures 8(a) and (b) illustrate the position and attitude responses during the 1m x -axis drag at optimal parameters. The DoFs other than the interacted axis are shown to maintain constant values throughout the duration of interaction. Insignificant changes in the attitude demonstrates the capability of a fully-actuated UAV, where the decoupled translational and rotational axes enable safe, intuitive interactions.

Optimization of a roll-axis drag demonstrates the novelty of the work in this paper and the ability of a fully-actuated drone to hold a non-zero roll setpoint without translating, as shown in Figure 9. This is another advantage of a fully-actuated drone over its underactuated counterpart, where interaction can occur in two additional DoFs (ie. roll and pitch). This allows for a greater range of applications, such as the need for tilted station keeping during physical interaction.

Similar trends as seen in Section 4.3 are observed across the z and yaw axes. Table 1 summarizes the optimal parameters found through simulations for the four DoFs and the corresponding reaction forces (F_r) and moments (τ_r). The reaction forces can be compared to the weight of the UAV (12.4N). The optimal x -drag and average z -drag reaction forces are 0.7% and 3.5% of the UAV's weight, respectively.

5 FREE FLIGHT TESTS

5.1 Test System

Figure 11 illustrates the free flight test system used. It consists of four main components:

	Parameter range, $M = 0$	Optimal $c_i (Ns/m)$	F_r, τ_r
x -drag	$c_x = 0.05 - 0.5$	0.17	0.085N
z -drag (up)	$c_z = 1.2 - 2.0$	1.62	0.49N
z -drag (down)	$c_z = 1.2 - 2.0$	1.27	0.38N
Roll-drag	$c_\phi = 0.3 - 0.6$	0.43	0.014Nm
Yaw-drag	$c_\psi = 0.01 - 0.5$	0.014	0.015Nm

Table 1: Summary of simulation optimal parameters obtained.

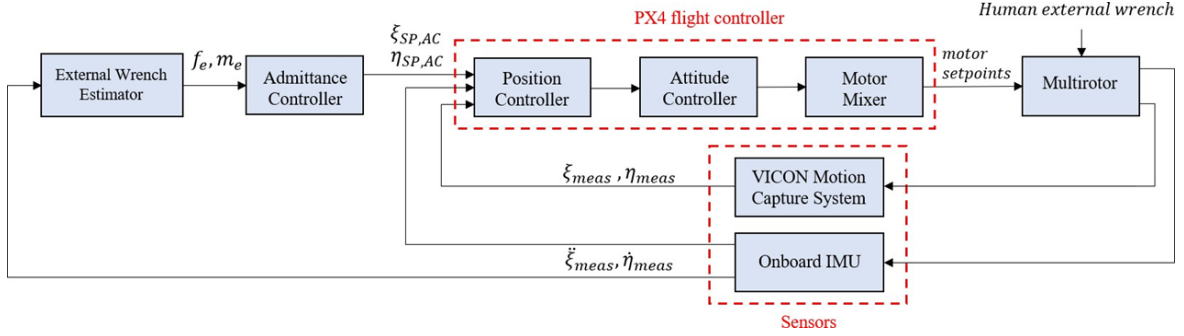


Figure 11: Experimental test system

1. The typical PX4 flight controller with cascaded P-PID controllers and the motor mixer.
2. The human external wrench directly applied by the human handler onto the UAV through physical contact.
3. Two types of sensing systems: the Vicon motion capture system with 12 cameras and an onboard IMU. The outputs of the two sensors are fused to compute the UAV's position and orientation information.
4. The sensor fusion, external wrench estimation and admittance controller setpoint computation are implemented and performed onboard the Pixhawk 4 mini control hardware.

Figure 12 depicts a handler performing PHDI in the motion capture lab.

5.2 Test Procedure

To validate simulation results, the following drag interactions were experimented and optimized with the following cases:

1. x -translational drag - $\xi_f = (0.5, 0, 0)m$
2. Yaw-drag - $\eta_f = (0, 0, 45)deg$
3. Combined interaction - $\xi_f = (0.75, 0, 1.5)m$

Note that testing for a roll-axis drag was not performed on the UAV used in this paper due to the tendency for the actuators to saturate with small moment applications. Additionally, due to the large spread of reaction forces at each damping coefficient in the z -axis testing, the optimization results have been omitted in this paper.

The dead band for each axis was first tuned to allow for any minor disturbances to be disregarded and to only update admittance controller setpoints once forces or moments were

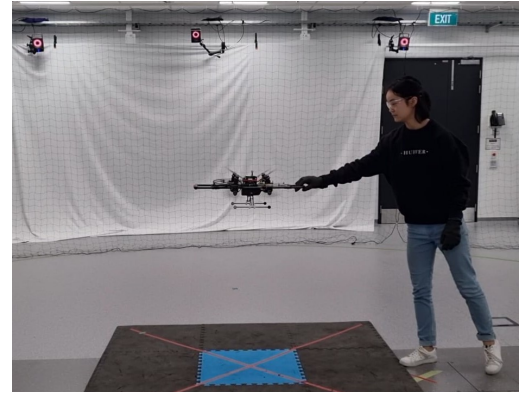


Figure 12: Free-flight test setup in motion capture lab with Vicon system

outside a specific range. The dead band also defines the duration of interaction. This was done by station keeping the UAV whilst logging the estimated forces and moments. Based on the residual estimated wrench, the dead band was determined for each axis.

Parameter m_i was gradually decreased from the default value. Zero m_i values were tested, however, this resulted in unstable and oscillatory behavior, thus, a non-zero value was selected for each of the axes. For each drag interaction experimented, different damping values were tested, and each combination of parameters was repeated 10 times to obtain averages. The lower bound was determined based on visual and log observations; significant oscillatory responses at low damping affected results and risked the safety of the handler, thus were omitted.

Table 2 shows the mean and maximum velocities encoun-

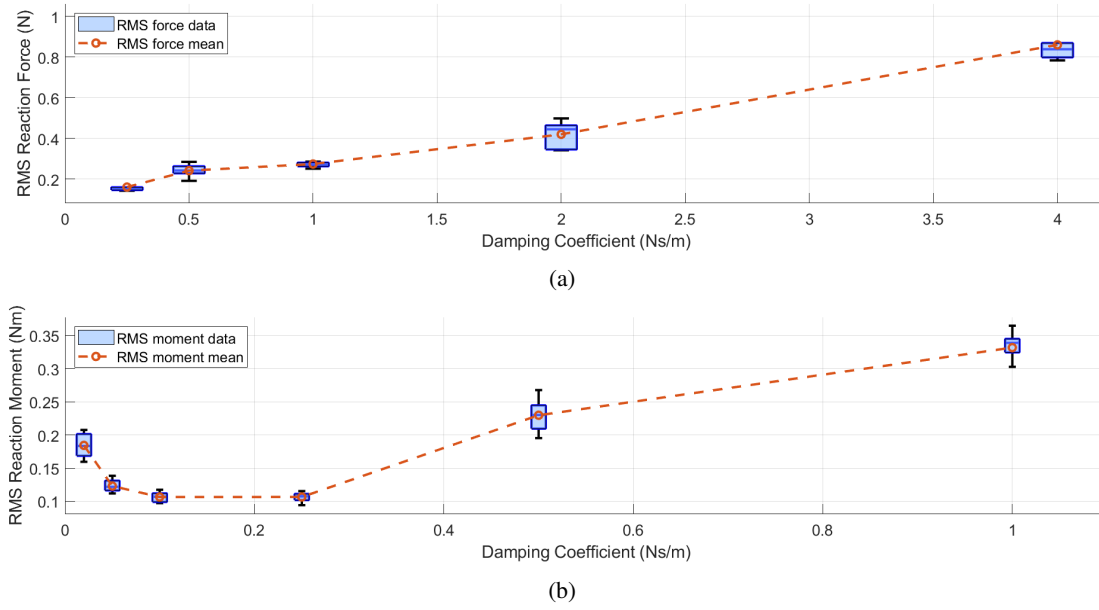


Figure 13: (a) Relationship between changing c_x with RMS reaction force for the x -axis drag in free flight tests at $m_x = 0.1$ and (b) relationship between changing c_ψ with RMS reaction moment for the yaw-axis drag in free flight tests at $m_\psi = 0.1$

tered during interaction in each axis. The maximum velocity recorded can therefore be quantified as the upper limit of the interaction rate for the sets of free flight tests performed in this study. Reaction forces and moments are directly related to the velocity during interaction, therefore consistent velocities are required to achieve repeatability.

	Mean velocity	Maximum velocity
x -drag	$0.21m/s$	$0.27m/s$
Yaw-drag	$0.29rad/s$	$0.39rad/s$

Table 2: Mean and maximum velocities during free flight tests for each tested axis

5.3 Results and Discussion

In Figures 13(a) and (b), the trends observed align well with those seen in simulation results. As the damping coefficient increases, the RMS reaction force and moment increase. The large spread observed in the data may be due to inconsistencies by the handler during interaction. In the x -axis drag tests, any c_x lower than $0.25Ns/m$ resulted in unsafe oscillatory behavior, thus results have been omitted. In the yaw-axis drag tests, an optimal point, $c_\phi = 0.1Ns/m$, is observed, as also depicted in the trends shown in simulations.

Table 3 summarizes the optimal parameters obtained through experimental testing. Compared to the weight of the UAV ($12.4N$), the optimal x -axis drag force is 1.3% .

Figure 14 shows the experimental data for a combined interaction where the UAV was physically interacted in multiple

	Parameter range	Optimal $c_i(Ns/m)$	F_r, τ_r
x -drag	$c_x = 0.25 - 4$	0.25	$0.16N$
Yaw-drag	$c_\psi = 0.02 - 1$	0.1	$0.11Nm$

Table 3: Summary of free flight optimal parameters obtained.

axes simultaneously, namely x -axis drag and z -axis drag. The shaded area on the graph depicts the region in which interaction occurred. The position appear to follow the setpoints well. At the end of the interaction, the position and attitude is held. A combined interaction shows that the axes are decoupled and a multi-axis interaction does not impact the UAV's ability to follow and hold setpoints.

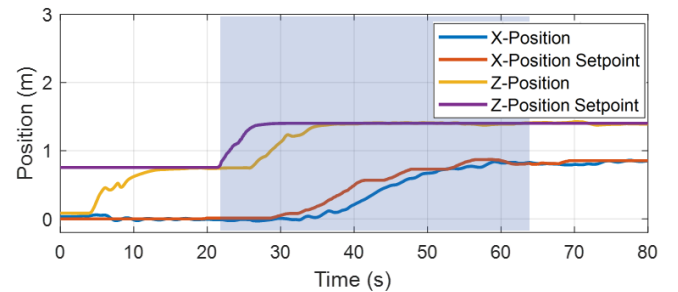


Figure 14: Combined interaction position graph using optimal parameters in free flight testing

6 CONCLUSIONS

This paper presents a novel analysis of admittance control on a compact, fully-actuated UAV for the purpose of human drone physical interaction. The main contribution is the optimization of the admittance controller in multiple translational and rotational axes to allow users to safely and intuitively interact with the UAV with minimal reaction forces and moments. The performance of the admittance controller was validated through simulations and experimental testing; optimal reaction forces of 0.7% and 1.3% of the UAV's weight were achieved for translation in the horizontal axis in simulations and free flight tests, respectively.

Future work will involve experimentally testing the roll and pitch-drag interactions to validate simulation results. Furthermore, an approach to differentiate between disturbance and human wrench will be investigated.

7 ACKNOWLEDGMENTS

The authors would like to acknowledge Pedro Mendes for his assistance and support for the work in this paper and the Drone Technology Research Group at the University of Auckland for their continuous feedback.

REFERENCES

- [1] Silvia Mirri, Catia Prandi, and Paola Salomoni. Human-drone interaction: state of the art, open issues and challenges. In *Proceedings of the ACM SIGCOMM 2019 Workshop on Mobile AirGround Edge Computing, Systems, Networks, and Applications*, pages 43–48, 2019.
- [2] Kei Nitta, Keita Higuchi, Yuichi Tadokoro, and Jun Rekimoto. Shepherd pass: ability tuning for augmented sports using ball-shaped quadcopter. In *Proceedings of the 12th International Conference on Advances in Computer Entertainment Technology*, pages 1–7, 2015.
- [3] Felix Huppert, Gerold Hoelzl, and Matthias Kranz. Guidecopter-a precise drone-based haptic guidance interface for blind or visually impaired people. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2021.
- [4] Jonas Auda, Nils Verheyen, Sven Mayer, and Stefan Schneegass. Flyables: Haptic input devices for virtual reality using quadcopters. In *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology*, pages 1–11, 2021.
- [5] Sujit Rajappa, Heinrich H Bühlhoff, Marcin Odelga, and Paolo Stegagno. A control architecture for physical human-uav interaction with a fully actuated hexarotor. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4618–4625. IEEE, 2017.
- [6] Sean Braley, Calvin Rubens, Timothy Merritt, and Roel Vertegaal. Griddrones: A self-levitating physical voxel lattice for interactive 3d surface deformations. In *UIST*, volume 18, page D200, 2018.
- [7] Jessica R Cauchard, Jane L E, Kevin Y Zhai, and James A Landay. Drone & me: an exploration into natural human-drone interaction. In *Proceedings of the 2015 ACM international joint conference on pervasive and ubiquitous computing*, pages 361–365, 2015.
- [8] Z Jeremy Chen, Jérémie XJ Bannwarth, Karl A Stol, and Peter J Richards. Analysis of a multirotor uav with tilted-rotors for the purposes of disturbance rejection. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 864–873. IEEE, 2018.
- [9] Markus Ryll, Giuseppe Muscio, Francesco Pierri, Elisabetta Cataldi, Gianluca Antonelli, Fabrizio Caccavale, and Antonio Franchi. 6d physical interaction with a fully actuated aerial robot. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5190–5195. IEEE, 2017.
- [10] Stéphane Caron, Abderrahmane Kheddar, and Olivier Tempier. Stair climbing stabilization of the hrp-4 humanoid robot using whole-body admittance control. In *2019 International conference on robotics and automation (ICRA)*, pages 277–283. IEEE, 2019.
- [11] Zhijun Li, Bo Huang, Zhifeng Ye, Mingdi Deng, and Chenguang Yang. Physical human–robot interaction of a robotic exoskeleton by admittance control. *IEEE Transactions on Industrial Electronics*, 65(12):9614–9624, 2018.
- [12] Federico Augugliaro and Raffaello D’Andrea. Admittance control for physical human-quadcopter interaction. In *2013 European Control Conference (ECC)*, pages 1805–1810. IEEE, 2013.
- [13] Arvid QL Keemink, Herman van der Kooij, and Arno HA Stienen. Admittance control for physical human–robot interaction. *The International Journal of Robotics Research*, 37(11):1421–1444, 2018.
- [14] Teodor Tomić, Christian Ott, and Sami Haddadin. External wrench estimation, collision detection, and reflex reaction for flying robots. *IEEE Transactions on Robotics*, 33(6):1467–1482, 2017.
- [15] PH Souza and Karl Stol. Analysis and design of a novel fully-actuated compact multirotor. In *Australasian Conference on Robotics and Automation (ACRA)*, 2021.
- [16] Christopher Banks, Antonio Bono, and Samuel Coogan. Physical human-uav interaction with commercial drones using admittance control. *IFAC-PapersOnLine*, 54(20):258–264, 2021.

Autopilot framework with INDI RPM control, real-time actuator feedback, and stability control on companion computer through MATLAB generated functions

Alessandro Mancinelli*, Erik Van Der Horst†, Bart D.W. Remes‡ and Ewoud J.J. Smeur§
Delft University Of Technology, Kluyverweg 1, Delft, NL

ABSTRACT

This paper presents a modular autopilot framework for Unmanned Aerial Vehicles (UAVs) that addresses the limitations of modern flight controllers. The framework utilizes separate and external subsystems for actuator control and the resolution of the Control Allocation problem. The actuator subsystem, implemented on a Teensy 4.0 microcontroller, incorporates an Incremental Dynamic Inversion Control (INDI) RPM controller, enabling direct control of motor RPM and facilitating the implementation of dynamic inversion-based control laws. The primary flight computer, a Cube Orange, coordinates the system, while an OrangePi 5 single-board computer serves as a companion computer. Real flight results demonstrate the effectiveness of the framework, highlighting its potential for robust and efficient UAV control.

1 INTRODUCTION

Unmanned Aerial Vehicles (UAVs), are aircraft designed to operate without a human pilot on board. UAVs have captivated a vast audience and found profound significance through their unmatched versatility, accessibility, and diverse range of applications across industries. From breathtaking aerial photography and aerial manipulation to efficient delivery and logistics.

In recent years, with growing interest from both industry and research communities, the field of Unmanned Aerial Vehicles (UAVs) has witnessed rapid advancements. The integration of a large number of actuators in UAVs has increasingly become prevalent, as exemplified by notable studies such as [1, 2, 3, 4]. The inclusion of an increasing number of actuators in UAVs serves a crucial role in achieving propulsion redundancy and enhancing overall performance. A prime example illustrating this is the dual-axis tilt rotor quad-plane depicted in Figure 1, where a total of 14 actua-



Figure 1: A picture of the dual-axis tilting rotor quad-plane developed at the TUDelft MAVLab.

tors, specifically 4 motors and 10 servomotors, are utilized to independently control the vehicle's 6 degrees of freedom.

Managing a significant number of actuators presents a complex challenge for modern commercial flight controllers, such as The Cube¹, Pixhawk 6c², Pixracer³ or the Lisa M⁴. These flight controllers offer limited Pulse-Width Modulation (PWM) actuator outputs, which are sometimes further restricted due to the shared PWM lines with auxiliary peripherals and sensors. Consequently, the integration of an external subsystem becomes indispensable to effectively handle the growing quantity of actuators in modern UAVs. Moreover, this external flight system must possess the capability to monitor the state of the actuators in real-time. The significance of precise real-time estimation of the actuator state becomes particularly evident in contemporary UAV design, where the Incremental Nonlinear Dynamic Inversion (INDI) control method is widely used[5, 6]. This advanced control method heavily relies on accurate actuator state estimation to ensure optimal performance. Traditionally, given the lack of actuator state feedback, the estimation of the actuator state is obtained by feeding the desired actuator input into an actuator model. The actuator model must be identified through laborious tests, which can introduce inaccuracies and consequently impact the quality of the estimation.

The ability to achieve reliable real-time measurements of the actuator state yields substantial advantages that extend beyond enhanced control performance. It assumes a pivotal role in promptly detecting actuator faults, facilitating the design

*a.mancinelli@tudelft.nl

†e.vanderhorst@tudelft.nl

‡b.d.w.remes@tudelft.nl

§e.j.j.smeur@tudelft.nl

¹<https://www.cubepilot.com/#/cube/features>

²https://docs.px4.io/main/en/flight_controller/pixhawk6c.html

³https://docs.px4.io/main/en/flight_controller/pixracer.html

⁴<https://wiki.paparazziuav.org/wiki/Lisa/M.v2.0>

and implementation of effective fault-tolerant control strategies. By possessing a dependable estimation of the actuator state, UAV systems can swiftly identify deviations or anomalies in actuator behavior, enabling timely interventions.

Furthermore, a notable constraint of commercial flight controllers lies in their limited computational power. Typically, these flight controllers utilize ARM Cortex M-family processors, which are specifically optimized for embedded applications. However, when compared to ARM A-family processors found in Single Board Computers (SBCs) like the Raspberry Pi [7], their computational power falls short. Embracing more powerful processors presents a promising opportunity to significantly enhance the computational capabilities of commercial flight controllers, enabling the execution of complex algorithms to support the UAV control and operations.

One potential way to improve the computational power of commercial flight controllers is through the adoption of an external companion computer, as demonstrated by previous works such as [8, 9, 10]. In particular, in our previous work [11], we utilized a Raspberry Pi 4B to solve the non-linear control allocation problem for the dual-axis tilting rotor quad-plane. This approach has proven to be highly successful and reliable.

In this paper, we present a modular autopilot framework that utilizes separate subsystems to address the aforementioned limitations of modern flight control. The primary flight computer employed is a Cube Orange, which runs the popular Paparazzi UAV autopilot software [12]. The main task of the primary flight computer is to estimate the vehicle states, run the autopilot routines and coordinate all the other sub-modules. As a companion computer sub-module, an OrangePi 5⁵ SBC was utilized, featuring a Rockchip RK3588S octa-core 64-bit processor. A notable feature of the OrangePi 5 SBC is its ability to run a Linux operating system, enabling the compilation and execution of functions developed in the MATLAB environment using the MATLAB C-coder toolbox⁶. For the actuator control and real-time feedback subsystem, a Teensy 4.0⁷ micro-controller was employed. The primary flight computer communicates with all the subsystems through custom modules developed in Paparazzi UAV, utilizing UART communication.

The paper is structured as follows: Section 2 presents the Modular Autopilot framework, which encompasses the Paparazzi UAV autopilot running on the primary flight computer. It also explores the information flow among all the subsystems and the primary flight computer. Section 3 provides a detailed analysis of the actuator control and real-time feedback subsystem, providing insights into its functionality and performance. Within this section, an INDI RPM controller is also presented. Section 4 focuses on the companion com-

puter sub-module and explores its setup for running MATLAB functions. Section 5 discusses and showcases real flight results obtained from the implemented framework. Finally, Section 6 draws conclusions based on the presented findings and discussions.

2 AUTOPILOT STRUCTURE

This section presents the Autopilot structure, covering both hardware and software aspects. Special attention is given to the working principles of Paparazzi UAV, the software running on the primary flight computer. The Paparazzi UAV flight software, running on the primary flight computer, is made available to the reader⁸.

2.1 Main Autopilot Hardware Structure

The overall Autopilot structure is illustrated in Figure 2. It can be observed that the AP structure consists of several parts. The main part is the Primary flight computer, where the estimation, stability, guidance, and navigation routines are implemented and executed. To achieve accurate estimation of flight states and position, the primary flight computer interfaces with multiple sensors, located both internally within the Orange Cube module and externally.

The internal sensors of the Orange Cube module include an Invensense ICM42688, an Invensense ICM20948, an Invensense MS5611 I2C barometer, and an AK099916 magnetometer. All the internal sensors are mounted on a temperature-controlled, vibration-isolated board.

For state estimation purposes, the primary flight computer utilizes also the following external sensors:

- UBLOX ZED-F9P RTK GPS connected through a Serial connection.
- Angle of Attack sensor employing a Megatron MAB12A magnetic encoder connected through PWM input.
- Sideslip angle sensor employing a Megatron MAB12A magnetic encoder connected through PWM input.
- External Drotek RM3100 magnetometer connected through the I2C bus.
- 4525DO Airspeed sensor employing a differential MS5611 barometer connected through the I2C bus.
- TFMini lidar with a maximum range of 15 meters, connected through the I2C bus.

All the sensor readings, except for the Sideslip and Angle of Attack readings are fused together using an Extended Kalman Filter based on the Estimation and Control Library (ECL) from PX4⁹, running on the Paparazzi UAV flight software.

⁵<http://www.orangepi.org/>

⁶<https://nl.mathworks.com/help/coder/index.html>

⁷<https://www.pjrc.com/store/teensy40.html>

⁸https://github.com/alessandro-mancinelli/Modular_AP_PPZ.git

⁹https://docs.px4.io/main/en/advanced_config/tuning_the_ecl_ekf.html

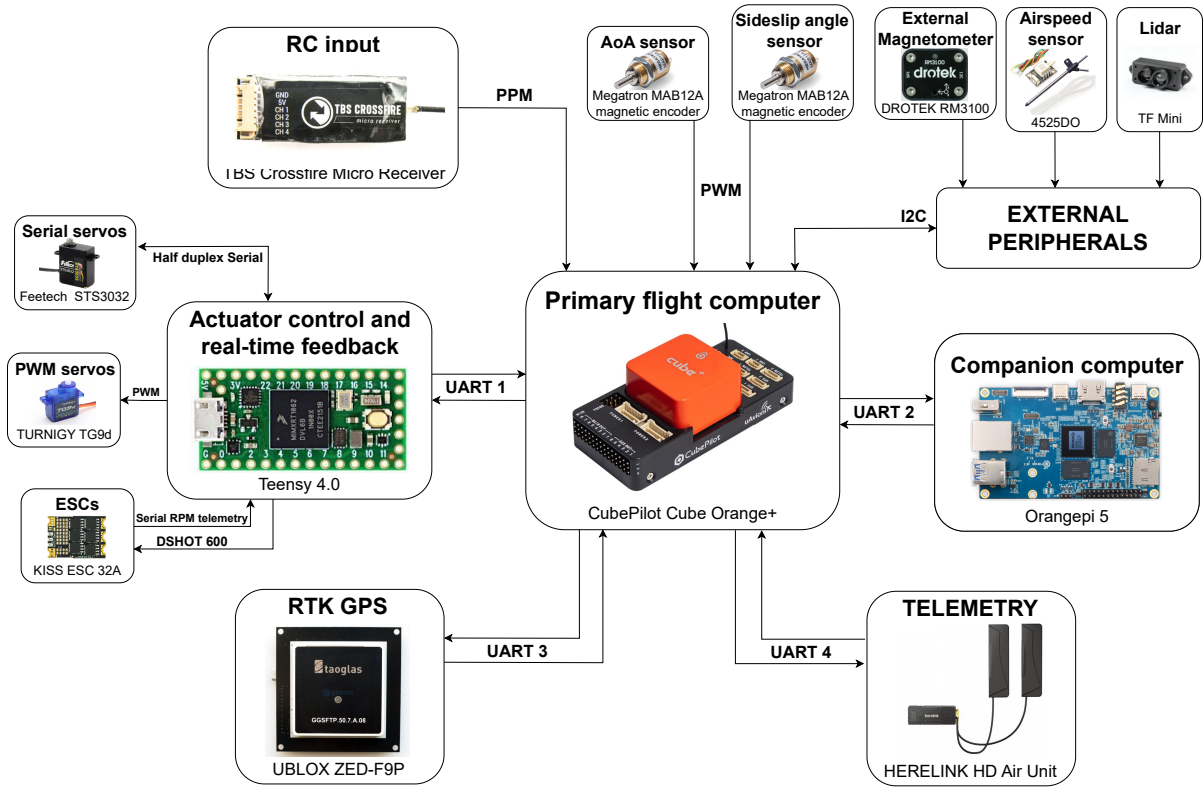


Figure 2: Hardware components and their intercommunication.

Regarding the Remote Control (RC) communication with the safety pilot of the UAV, an RC input source is established using a TBS Crossfire Micro receiver connected through a PPM signal. Additionally, a Herelink HD Air unit connected through UART is utilized to establish communication with the Ground Control Station.

A serial communication is utilized for the communication between the Primary Flight Computer and both the Actuator control and the real-time feedback subsystem, and the Companion computer subsystem. In our specific case, the companion computer is responsible for solving the control allocation problem in real-time. Since the solution needs to be communicated to all the actuators, this requires a fast and reliable communication channel. To minimize overhead, we implemented a UART communication protocol with one starting byte and one checksum byte. The baud rate chosen for the Serial communication is 1.5 megabaud.

2.2 Paparazzi UAV flight software

Paparazzi UAV is a versatile and highly customizable flight software that offers users a wide range of options for configuring their aircraft. At the core of the Paparazzi software is the airframe XML file, which serves as the central configuration hub for selecting and setting up all the necessary components and modules required for a successful flight. This modular approach empowers users to customize

their aircraft according to specific requirements by choosing the appropriate sensors, actuators, and other peripherals. A comprehensive overview of the information flow in Paparazzi UAV is available on the wiki ¹⁰.

One of the key strengths of Paparazzi UAV is its support for custom modules, which facilitate communication and collaboration among different components. These modules can interact with each other using external global variables or, in a more structured manner, through an intermediary routine known as the Application Binary Interface (ABI) routine. This enables seamless integration and coordination between various functionalities, enhancing the overall capabilities and adaptability of the system.

2.2.1 The ABI publisher publish/subscribe middle-ware

ABI is a custom publish/subscribe middleware that facilitates the exchange of data between software components. Within the Paparazzi UAV software, two different modules can exchange information through ABI. Once the ABI message structure is defined, each module can utilize the ABI routine as either a publisher or a subscriber of information. The role of a publisher is to update the content of the ABI node with new information to be broadcasted in the autopilot. One or

¹⁰<https://wiki.paparazziuav.org/wiki/DevGuide/DesignOverview>

more modules can then subscribe to that ABI node. An interesting aspect is that the subscriber node is notified whenever the ABI message contains fresh data and automatically triggers a function to post-process the data. In the design of the Modular Autopilot framework, we extensively utilized the ABI middleware. Specifically, we associated an ABI message with both the Actuator Control and Real-time Feedback subsystem and the Companion Computer subsystem, enabling interaction between our control module and these subsystems.

2.2.2 Custom Teensy actuators Paparazzi module

The custom module called `serial_act.t4`, developed within the Paparazzi UAV software, serves as the software back-end running on the Primary Flight Computer. Its purpose is to enable communication between the Actuator Control and Real-time Feedback subsystem running on the Teensy 4.0 and the Primary Flight Computer. The messages exchanged through the Serial UART channel between the Teensy 4.0 and the Primary Flight Computer are defined in the `ca.am7.h` file, which is located in the "Modular_AP.PPZ/sw/airborne/modules/sensors" path. It should be noted that the "Modular_AP.PPZ" refers to the Paparazzi repository available at the TUDelft github page⁸. In the `serial_act.t4.h` file, two structures are defined: `serial_act.t4_data_in` and `serial_act.t4_data_out`, representing the inbound and outbound messages, respectively. In our case, we have decided to include a rolling message with lower priority in the main message struct. Whenever a message is sent, one element of a float array with a length of 255 is also transmitted. Consequently, the update rate of the float array is approximately 0.4% of the refresh rate of the main message.

This module is associated with a publishable ABI message named "SERIAL_ACT_T4.OUT". Users can publish to this message from any location within the Paparazzi software, allowing the outbound message structure to be sent over the hardware serial interface to the Teensy.

Similarly, an ABI broadcast message called "SERIAL_ACT_T4.IN" is triggered whenever a new inbound message structure is received from the Teensy. This broadcast message can also be accessed from any location within the Paparazzi software.

To gain a comprehensive understanding of how to subscribe to or publish an ABI message, readers are encouraged to consult the Paparazzi UAV ABI documentation¹¹

2.2.3 Custom Orangepi PPZ module

Similarly to the implementation of the `serial_act.t4` module, a custom module called `ca.am7` module was developed within the Paparazzi UAV software as a back-end

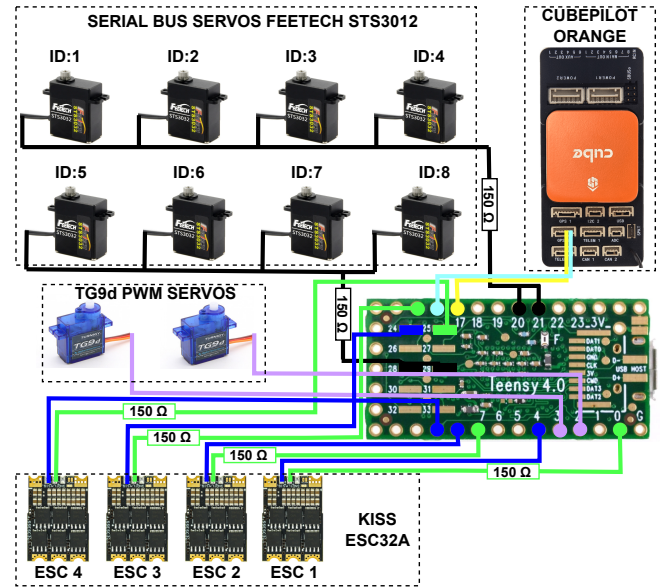


Figure 3: Electrical schematic of the actuator control and real-time feedback subsystem. The $150\ \Omega$ resistors are utilized to safeguard the bus from potential transmission conflicts.

running on the Primary flight computer to enable communication with the OrangePi 5. The inbound and outbound message structures exchanged with the OrangePi are defined in the `ca.am7.h` file, located in the "Modular_AP.PPZ/sw/airborne/modules/sensors" path. Just like the `serial_act.t4` module, we also included a rolling message with lower priority in the main message struct.

The ABI broadcast message containing the inbound data structure from the OrangePi is named "AM7_IN", while the publishable ABI message for the outbound data transmission is named "AM7_OUT".

3 ACTUATOR CONTROL AND REAL-TIME FEEDBACK SUBSYSTEM

In this section, we present the code of the Actuator control and real-time feedback subsystem running on the Teensy 4.0. We also provide a brief overview of the protocol used to control both the serial servos, PWM servos, and the ESCs. The source code for generating the binary file for the Teensy 4.0 is available in the `Teensy_actuators.PPZ` GitHub repository¹². The software is developed using the Arduino IDE and the `Teensyduino` add-on⁷. The electrical scheme of the actuator control and real-time feedback subsystem is depicted in Figure 3.

3.1 Serial bus servos control

In the current configuration, the Teensy 4.0 microcontroller controls a total of 8 Feetech STS3012 serial servos.

¹¹https://paparazzi-uav.readthedocs.io/en/stable/developer_guide/abi.html

¹²https://github.com/tudelft/Teensy_actuators.PPZ.git

Header	ID number	Data Length	Command	Parameter	Checksum
0x55 0x55	ID	Packet Length	Cmd code	Prm 1... Prm N	Checksum

Table 1: Command packet frame for the communication between the Teensy 4.0 and the Feetech STS3012 servos.

These 8 servos are controlled using two separate buses, with each bus connected to 4 servos. The protocol used to communicate with the serial servos is a single-wire half-duplex serial protocol. It involves standardized packet transmission between a master device (the Teensy 4.0) and slave devices (the STS3012 servos). An overview of the packet frame is provided in Table 1.

The master device sends an instruction packet on the bus, addressing a specific servo ID. All devices on the bus receive the packet, but the servos discard it if their ID does not match. Only the servo with the matched ID executes the instruction and then sends a confirmation packet.

The servo ID is a value ranging from 0 to 253, which means that a maximum theoretical number of 254 servos per bus is supported. However, since the bus is shared among all servos and the bus bandwidth is limited to 1 Mega baudrate, increasing the number of servos will limit the refresh rate of the instructions that can be sent to each servo.

Furthermore, it should be noted that if we want to both command a servo position and obtain servo position feedback, two instruction packets have to be sent to the servo. The first instruction contains the desired servo position, to which the servo will respond with an ack packet. Then, we need to send another instruction requesting the servo's current angular position, to which the servo will respond with a packet containing its current position. This can be extended to include additional information such as load, tension, or angular speed. For a detailed explanation of all the possible commands for the serial servos, the reader can refer to the Documentation/Serial_servo_protocol.pdf document in the Teensy_actuators_PPZ GitHub repository¹².

We observed that typically each servo takes 20 to 50 microseconds to respond to a Teensy request. Due to this response time and our requirement for high-speed position control and feedback from the servos, we decided to use two separate buses to control the 8 servos. This approach allowed us to achieve a refresh rate of 350 Hz for both the angular position control and angular position feedback of each servo, while providing enough time to avoid bus conflicts. As an additional safety feature, we also added a 150 Ω resistor to the serial line. This resistor helps limit the maximum current flow on the Teensy serial pin to 22 mA in the event of conflicts on the serial line. Without the resistor, the current flow on the Teensy serial pin in the case of a conflict would exceed the maximum limit of 25 mA tolerated by the microcontroller, potentially resulting in chip damage.

3.2 PWM servos control

On top of controlling Serial bus servos, the Teensy_actuators_PPZ software running on the Teensy 4.0 can also control conventional PWM servos. In our current setup, we have utilized two PWM TGYd micro servos, which are assigned to the ailerons of the airframe. These servos, due to their communication technology, do not feature any feedback regarding their angular position. Therefore, an estimation of the servo position is carried out on the Teensy board. The estimation is based on a first-order transfer function that incorporates the command and the user-provided first-order dynamics characteristics of the servo. These parameters can be identified through a step response test using external equipment, such as an IMU mounted on the servo arm, as demonstrated in [3].

The routine responsible for PWM servo control and estimation is implemented in the `writeEstimatePwmServos()` function, which can be found in the `servo_esc_control_w_feedback_T4_PPZ.ino` file in the Teensy_actuators_PPZ GitHub repository¹².

3.3 Control of the ESCs

The KISS ESC 32A in our system are controlled using Digital Shot (DSHOT) commands at a speed of 600 bit/s. The DSHOT protocol has become widely adopted as a digital communication protocol specifically designed for controlling ESCs. It offers several advantages over conventional UART communication.

One notable advantage of the DSHOT protocol over the UART protocol lies in how zeros and ones are identified in the communication. In DSHOT, the differentiation between zeros and ones is based on the duration of the high state voltage, rather than relying on traditional binary representation through voltage levels. This utilization of timing information allows for more precise encoding of data.

The DSHOT packet frame consists of a total of 16 bits. The initial 11 bits are dedicated to identifying the throttle command. A value of all zeros indicates that the motor is disarmed, while values 1-47 are reserved for special purposes. Consequently, the throttle command can range from 0 to 2000. Following the 11-bit throttle command, one bit is allocated for telemetry request, and an additional 4 bits contain the cyclic redundancy check (CRC) for message integrity.

When the telemetry bit is enabled, the ESC outputs a telemetry packet through a dedicated serial line operating at a baud rate of 115200. The telemetry packet comprises 10 bytes of information, including temperature, voltage, current, and RPM data from the ESC. For a more detailed explanation

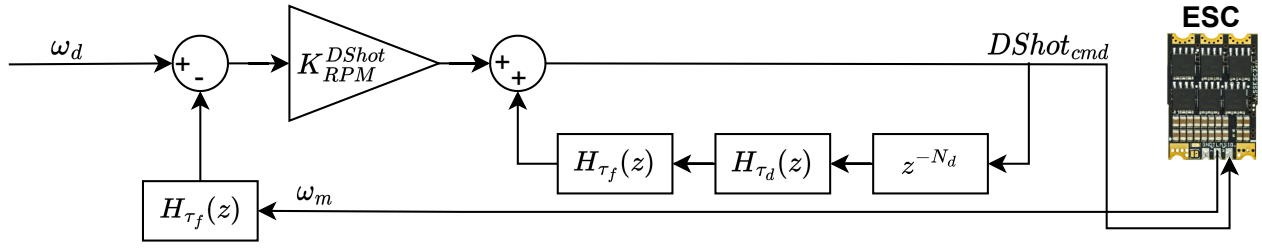


Figure 4: Scheme of the RPM INDI controller.

of the telemetry protocol, the reader can refer to the Documentation/KISS_ESC_telemetry_protocol.pdf file available in the GitHub repository¹².

Due to the limited baud rate of this serial line, we decided to employ a refresh rate of 500 Hz for both DShot throttle commands and telemetry transmission. Keeping within this limit ensures reliable and timely telemetry updates from the ESCs.

3.3.1 Motor RPM control

The high-speed command and telemetry capabilities of the ESCs have allowed for the design and implementation of an RPM controller for the motors. The ability to reliably control the motor's RPM is a crucial feature for modern controllers that rely on model inversion to compute the actuator solution. By directly setting the actuator RPM, the need to accurately model external factors such as battery discharge or propeller inflow is eliminated. This simplifies the control process and enhances the overall performance of the system.

The INDI RPM controller scheme used to control the motor RPM is illustrated in Figure 4, where ω_m is the RPM value measured by the ESC and ω_d is the desired RPM. The transfer function $H_\tau(z)$ represents a low-pass filter in the discrete-time domain with a time constant of τ :

$$H_\tau(z) = \frac{\tau}{z - (1 - \tau)}, \quad (1)$$

where z denotes the discrete-time step delay operator.

Through experimentation, we determined that this filter with a time constant of $\tau_f = 0.1131$ effectively filters out the noise present in the ESC readings. With the ESC telemetry frequency of 500 Hz, this corresponds to a first order low-pass filter with a cutoff frequency of 60 rad/s in the Laplace domain.

Another filter of the form of Equation 1 with parameter τ_d represents the first-order dynamics of the motor system, while N_d in Figure 4 represents the discrete-time steps of delay introduced by the ESC for processing the DShot command. To determine these parameters, we analyzed the step response for various DShot commands. The analysis of the system response, as shown in Figure 5, yielded an estimated value of

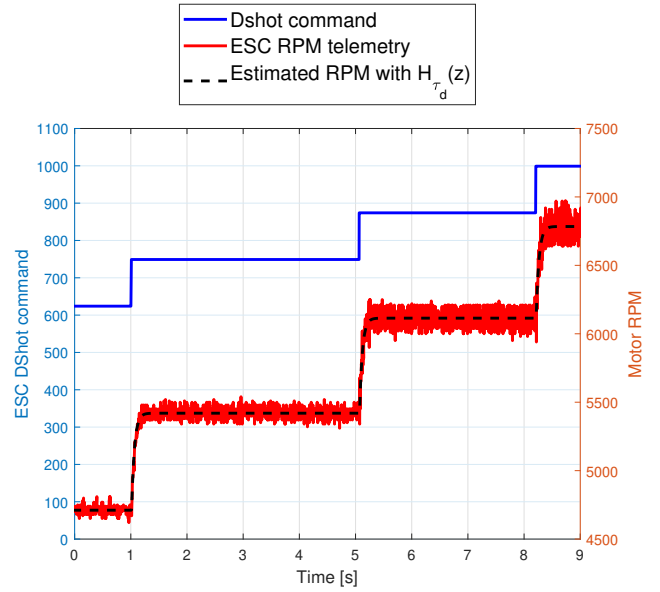


Figure 5: Identification of the motor dynamics.

0.039 for τ_d and 3 for N_d . It is important to note that these values are specific to the tested set of propellers and to an RPM control loop update rate of 500 Hz, matching the ESC telemetry refresh rate.

The gain K_{RPM}^{DShot} , on the other hand, represents the inverse of the control effectiveness, in this case the mapping from RPM to DShot commands. We opted for a constant gain of $K_{RPM}^{DShot} = 0.096$, which corresponds to the ratio between the maximum RPM and the maximum DShot command.

To illustrate the performance of the INDI RPM controller, a step response from an initial RPM of 5000 to a desired RPM of 10000 is depicted in Figure 6.

3.4 Communication with the Primary Flight Computer

For communication with the serial_act.t4 module in Paparazzi UAV presented in Section 2.2.2, the Teensy also utilizes UART communication using the same inbound and outbound structures defined in the serial_act.t4_data.in and serial_act.t4_data.out data structures. These data structures can be found in the Definitions.h file in the Teensy_actuators_PPZ

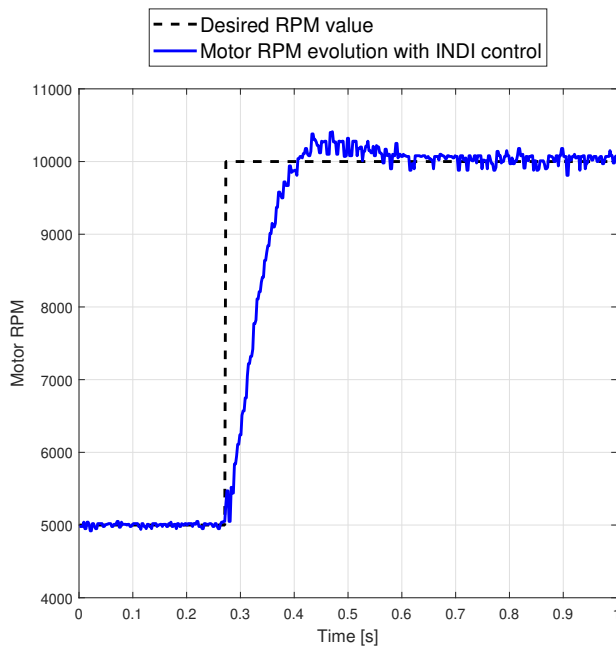


Figure 6: RPM step response using the INDI RPM controller.

GitHub repository¹².

It is crucial to ensure that the structures defined in the Paparazzi module and in the Teensy firmware are identical. Any inconsistencies between the structures will result in improper communication between the hardware modules.

4 COMPANION COMPUTER SUBSYSTEM

As mentioned in the introduction, the utilization of ARM-A family processors in conjunction with the embedded ARM-M family processor significantly enhances the computational capabilities of any robotic platform. In our specific implementation, we employed an OrangePi 5 to tackle the complex Nonlinear Control Allocation problem for the dual-axis tilting rotor quad-plane.

The communication with the custom `ca_am_7` module running on the Primary Flight Computer through UART is implemented on a separate thread on the OrangePi 5. Unlike the Teensy 4.0, which has a single-core processor, the OrangePi 5 features an Octa-Core processor. This enables us to distribute the serial communication routine and the control routine across different threads running on different cores.

The software developed for this purpose is available in the GitHub repository `OrangePi_PPZ`¹³. The software is structured around the `am7x.h` and `am7x.c` files. The `am7x.h` file contains the inbound and outbound data structures for serial communication with the Primary Flight Computer. As mentioned in the previous section, it is crucial for these data structures to match those implemented in the Paparazzi UAV `ca_am_7` module, as presented in Section 2.2.3.

¹³https://github.com/tudelft/OrangePi_PPZ

The `am7x.c` file implements UART communication within one thread and executes the MATLAB-generated function (or any other C function useful for the autopilot) on a separate thread. In our case, we execute a C-function automatically generated by the MATLAB C-coder toolbox. For information on generating a C function from a MATLAB function, please refer to the MATLAB documentation⁶.

All the MATLAB-generated files are located in the `MATLAB_generated_files` folder of the `OrangePi_PPZ` repository¹³ and can be invoked from the `am7x.c` file.

5 FLIGHT TEST RESULTS

To demonstrate the efficacy of the proposed Autopilot framework, we conducted tests using the dual-axis tilting rotor quad-plane depicted in Figure 1. During the flight, the companion computer subsystem computed the actuator solution, which was then transmitted to the actuator control and real-time feedback subsystem. Figure 8 presents a photograph of the Autopilot hardware, while Figure 7 illustrates the observed evolution of the actuators command and state during the flight test.

6 CONCLUSION

In this paper, we introduced a cutting-edge modular Autopilot framework designed to handle a multitude of actuators while providing real-time, high-frequency feedback on their state. We presented comprehensive software and hardware details necessary for implementing the framework on a wide range of UAVs using the Paparazzi UAV platform. We have also developed an INDI RPM controller leveraging the high-frequency feedback, enabling precise control of motor RPM. Furthermore, the modular Autopilot includes a companion computer subsystem, capable of efficiently solving complex real-time tasks to support the Primary Flight Computer. The effectiveness of the Autopilot functionalities was then demonstrated through a successful flight test of a dual-axis tilting rotor quad-plane equipped with the system.

ACKNOWLEDGEMENTS

The work was carried out within the Unmanned Valley Project. The authors would like to thank the "Europees Fonds voor Regionale Ontwikkeling(EFRO)" who is founding the Unmanned Valley project under grant code KvW-00168 for the South-Holland region.

REFERENCES

- [1] C.DeWagter, B.Remes, E.Smeur, F.VanTienen, and R.Ruijsink. The nederdrone: A hybrid lift, hybrid energy hydrogen uav. *International Journal of Hydrogen Energy*, Volume 46, Issue 29, 2021.
- [2] Ali Bin Junaid, Alejandro Diaz De Cerio Sanchez, Javier Betancor Bosch, Nikolaos Vitzilaos, and Yahya Zweiri. Design and implementation of a dual-axis tilting quadcopter. *Robotics*, 7(4), 2018.

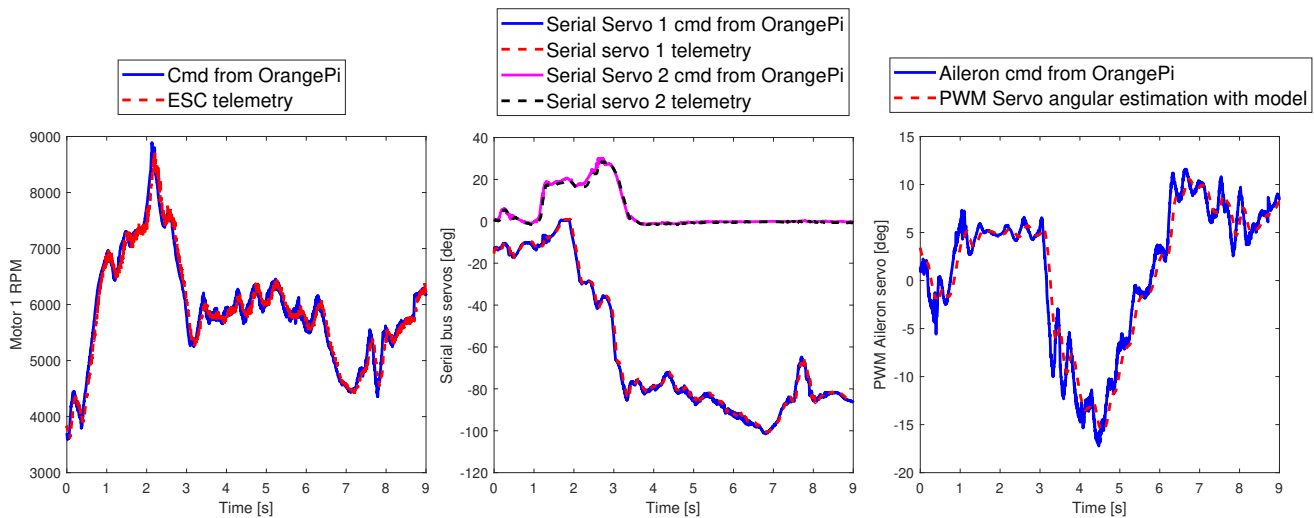


Figure 7: Actuators evolution during a portion of the dual-axis tilting rotor quadplane flight test.

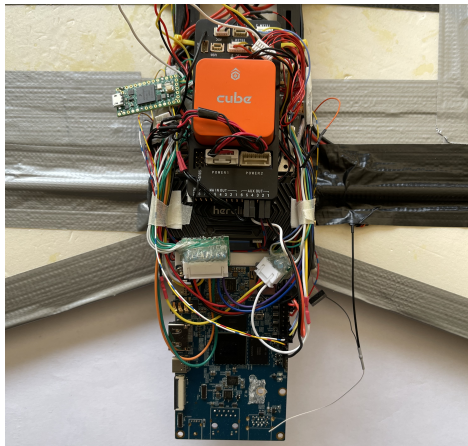


Figure 8: A picture of the Autopilot hardware mounted on the dual-axis tilting rotor quad-plane.

- [3] A. Mancinelli, E.J.J. Smeur, B. Remes, and G.D. Croon. Dual-axis tilting rotor quad-plane design, simulation, flight and performance comparison with a conventional quad-plane design. *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2022.
- [4] Christos Papachristos, Kostas Alexis, and Anthony Tzes. Efficient force exertion for aerial robotic manipulation: Exploiting the thrust-vectoring authority of a tri-tiltrotor uav. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4500–4505, 2014.
- [5] E.J.J. Smeur, G.C.H.E. de Croon, and Q. Chu. Cascaded incremental nonlinear dynamic inversion for mav distur-

bance rejection. *Control Engineering Practice*, 73:79–90, 2018.

- [6] Gabriele Di Francesco, Massimiliano Mattei, and Egidio D’Amato. Incremental nonlinear dynamic inversion and control allocation for a tilt rotor uav. *AIAA Guidance, Navigation, and Control Conference*, 2014.
- [7] Gareth Halfacree Eben Upton. *Raspberry Pi User Guide*. Wiley, 2016.
- [8] J. Yang, A.G. Thomas, S. Singh, S. Baldi, and X. Wang. A semi-physical platform for guidance and formations of fixed-wing unmanned aerial vehicles. *MDPI sensors*, 2020.
- [9] Gustavo Gargioni, Marco Peterson, J. B. Persons, Kevin Schroeder, and Jonathan Black. A full distributed multipurpose autonomous flight system using 3d position tracking and ros. *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1458–1466, 2019.
- [10] Hannah Mohr. Uav implementation of distributed robust target location in unknown environments. *2020 IEEE Aerospace Conference*, pages 1–10, 2020.
- [11] Alessandro Mancinelli, Bart D. W. Remes, Guido C. H. E. De Croon, and Ewoud J. J. Smeur. Real-time nonlinear control allocation framework for vehicles with highly nonlinear effectors subject to saturation. *Journal of Intelligent & Robotic Systems*, 2023.
- [12] B. Gati. Open source autopilot for academic research - the paparazzi system. *2013 American Control Conference*, 2013.

Cascaded Indoor Flight Controller Design for a Miniaturized Tiltwing Aircraft

J. Müller*, D. Schatten, D.F. Duda, and D. Moormann
RWTH Aachen University, Templergraben 55, 52062 Aachen Germany

ABSTRACT

This paper presents a concept for indoor operation of a tiltwing aircraft based on a cascaded controller for translational velocity commands. We outline the challenges of flying a tiltwing aircraft indoors, especially with regard to navigation and confined space requirements. By using Incremental Nonlinear Dynamic Inversion (INDI), we are able to control our micro tiltwing aircraft in translational direction with respect to a horizontal reference frame. Here, focus is placed on modeling the translational control effectiveness considering aerodynamic and propulsion models. The control performance is analyzed by means of simulation.

1 INTRODUCTION

Operating unmanned aerial systems in confined spaces, including indoor settings, offers a number of opportunities, including remote aerial surveillance, inspections, and data collection. However, such an operation environment involves various challenges, especially when using tiltwing aircrafts. These hybrid, unmanned vehicle have the ability to vertically take off and land (VTOL) and also fly in a fast, efficient forward flight. Accordingly, these systems require different airspace dimensions for safe operation, depending on the desired airspeed.

In this paper, we present a concept for controlling a miniaturized tiltwing aircraft for indoor operations. Initially, a dynamic aircraft model, conceptional constraints for controlling a tiltwing aircraft in confined spaces and the indoor position and heading determination are presented. Afterwards, the translational controller implementation is introduced with focus on the application of the INDI control law on the aircraft's translational motion. Since the examined aircraft naturally rotates its wing and operates in a low Reynolds number range, instationary effects occur, especially during the transition from thrust- to wing-borne flight. This leads to challenges in creating an aerodynamic model of the aircraft, which is required for the controller design. We address this issue by using a translational INDI

controller, which, due to its robustness, allows for simplified aerodynamic modeling and compensates for model inaccuracies. The modeling of this control concept is described in Section 3, along with other required controller cascade. Simulation results for the implemented controller are presented in Section 4 and a conclusion is drawn in Section 5.

2 INDOOR FLIGHT

The tiltwing aircraft on which this study is based is shown in Figure 1 and is further referred to as *Idefix*. With a span of $b = 0.5$ m and a mass of $m = 175$ g, it was especially designed for indoor operations.



Figure 1: The tiltwing aircraft *Idefix*

The control devices of *Idefix* are shown in Figure 2. In thrust-borne configuration, the main motors positioned at the wing provide lift and roll control, while a tail-mounted motor is used for pitch control and yaw control is provided by aileron deflections.

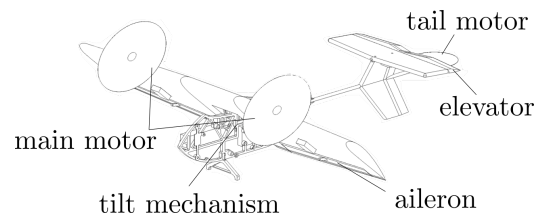


Figure 2: Control devices of the tiltwing aircraft.

*Email address: julian.mueller@fsd.rwth-aachen.de

With increasing longitudinal velocity, the aircraft continuously transitions from thrust- to wing-borne flight, and the main wing rotates along the lateral axis. This process is further referred to as *transition*. In wing-borne configuration, the tiltwing aircraft is actuated like a conventional aircraft except for the yaw-control, which mainly originates from differential thrust instead of a rudder.

2.1 Dynamic model description

The aircraft's translational motion is described by following equation:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} \dot{u}_F \\ \dot{v}_F \\ \dot{w}_F \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad (1)$$

Here, $[u \ v \ w]^T$ describes the translational velocity and $[p \ q \ r]^T$ the rotational velocity in body fixed coordinates, see Figure 3. The term $[\dot{u}_F \ \dot{v}_F \ \dot{w}_F]^T$ considers the acceleration resulting from external forces acting on the aircraft's center of gravity. As shown in Figure 3, these forces include the thrust force $\vec{T}$, which is aligned with the wing-chord and tilted by σ towards the aircraft's longitudinal axis, the aerodynamic force $\vec{A}$ resulting from the airspeed and the gravitational force $\vec{G}$. Equation 2 represents these forces in body-fixed coordinates.

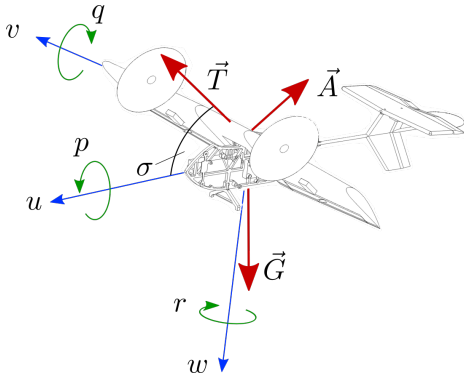


Figure 3: Direction of body fixed translational, angular velocities and acting forces.

$$\begin{bmatrix} \dot{u}_F \\ \dot{v}_F \\ \dot{w}_F \end{bmatrix} = \begin{bmatrix} \frac{T \cdot \cos(\sigma)}{m} - \frac{G}{m} \cdot \sin(\Phi) - \frac{A_x}{m} \\ \frac{G}{m} \cdot \cos(\Theta) \cdot \sin(\Phi) - \frac{A_y}{m} \\ -\frac{T \cdot \sin(\sigma)}{m} - g \cdot \cos\Phi \cdot \cos(\Theta) - \frac{A_z}{m} \end{bmatrix} \quad (2)$$

The angular motion is described in Equation 3 with $[\dot{p} \ \dot{q} \ \dot{r}]^T$ being the body-fixed rotational accelerations, I representing the inertia tensor, and $[M_x \ M_y \ M_z]$ covering all external moments such those generated by the motors or the deflection of the aerodynamic control surfaces.

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = I^{-1} \left(\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \left(I \cdot \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right) \right) \quad (3)$$

For simulations, the aerodynamic forces and moments are modeled using an element-based approach. Structural parts, such as the wing, control surfaces and fuselage are subdivided into sections with associated drag and lift polars. The associated forces and moments can be derived by calculating the inflow at each element. For elements exposed to inflows with an angle of attack $\alpha > 12^\circ$, the aerodynamic polars are extrapolated according to [1]. Each element's associated aerodynamic forces and moments are then superimposed in the aircraft's center of gravity [2]. Flight tests with various tiltwing aircraft have shown that the described approach models the aerodynamics with sufficient accuracy.

2.2 Transition flight in confined space

Flight systems are restricted by specific translational acceleration and deceleration limits, as well as by certain flight attitudes in order to operate safely. These aircraft-specific limitations directly affect the required flight space for defined maneuvers. Thus, the proximity to obstacles, such as walls and ceiling, pose significant limitations that must be considered in mission planning and controller architecture for autonomous indoor operation. Unlike thrust-borne multi-copters, which have similar limitations in all directions, a transition-capable aerial system like *Idefix* has direction-specific limitations that pose a special challenge in controlling these vehicles.

As shown in Figure 4, tiltwing aircrafts accelerate continuously along the body-fixed longitudinal axis, building up a desired airspeed while tilting the wing downward to the fixed-wing position. In the process, the airspeed increases, allowing the wing to generate more lift. To maintain altitude, the weight force must be in balance with the lift and thrust forces. In doing so, the controller must account for nonlinear limitations of the flight system, such as aerodynamic stall. In contrast to the transition of tiltwing aircrafts in outdoor operations, the influence of wind can be neglected indoors. This allows to estimate the aerodynamic inflow only based on the direction of translational motion and the attitude angles of the aircraft. In addition to these aerodynamic limitations, the absolute deflection of the actuators, such as the elevators and ailerons and the thrust magnitude from the propellers, are limited. Alongside these static limitations, the actuators are subjected to dynamic limitations, including the angular velocity of the tilt mechanism. As some of these actuators act only in specific directions, these limitations are directional-dependent.

To approximate the required transition length along a straight line, a constant translational acceleration of $\ddot{u} = 1 \text{ m/s}^2$ is assumed. Simulations show, that for *Idefix* a sufficient wing-borne airspeed is reached at $u = 7 \text{ m/s}$. Using these boundary conditions, the double integration of

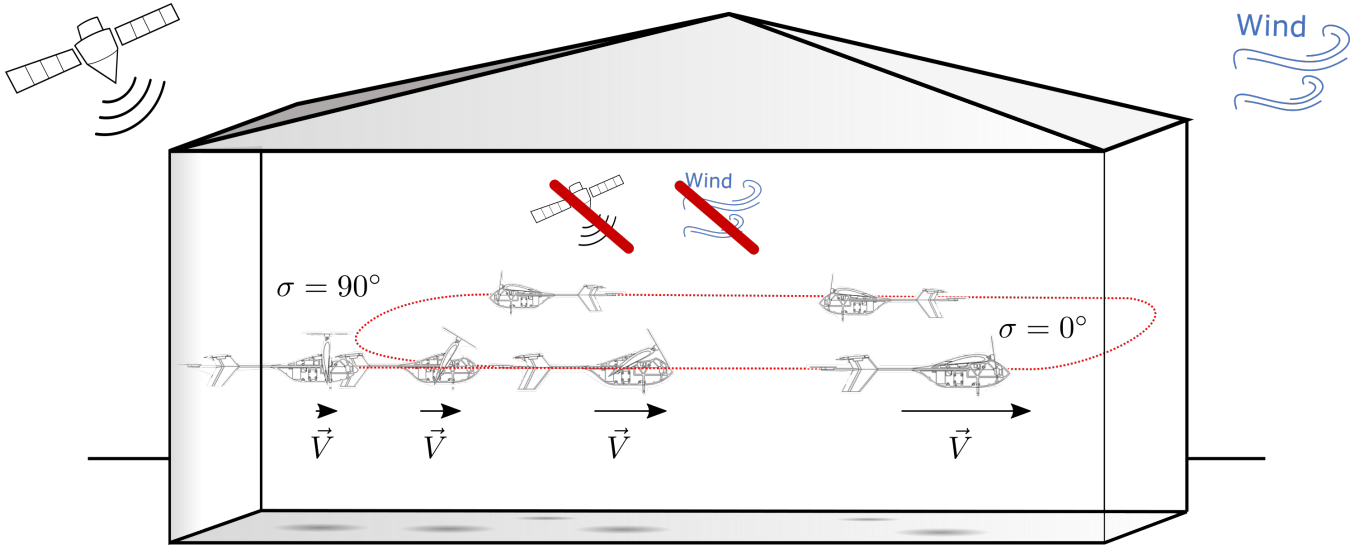


Figure 4: Indoor transition from thrust- to wing-borne flight

Equation 1 yields a transition length of $l_{tr} \approx 25$ m. Thus, to fully transition from thrust-borne to wing-borne flight and back with $\dot{u} = \pm 1 \text{ m/s}^2$, a minimum room length of $l \approx 50$ m is required. This considerably restricts the range of indoor operation for these tiltwing system.

To allow for wing-borne operation in smaller areas, the space requirement for transitions along a circular path is evaluated. Assuming no external disturbance, such as no prevailing wind, the minimum curve radius R_{min} can be calculated depending on the horizontal ground speed V_k and lateral acceleration a_y , which during turns is dependent on the aircraft's roll angle [3]. With Equation 4 and Equation 5, the minimum curve radius for wing-borne flight can be determined to $R_{min} = 7.5$ m.

$$R_{min} = \frac{V_k^2}{|a_y|} \quad (4)$$

$$a_y = \tan(\phi) \cdot g \quad (5)$$

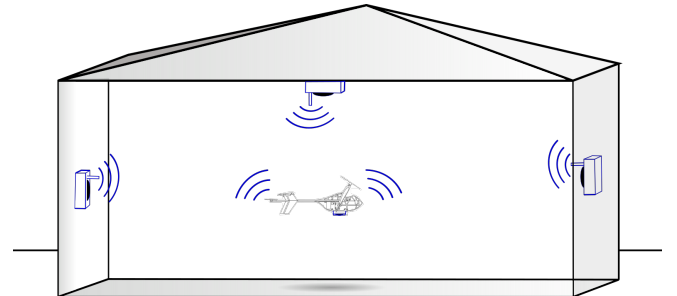
This presumes a maximum roll angle of $\phi = 45^\circ$ and an increased ground speed of $v = 8.5 \text{ m/s}$, which accounts for the extra lift required during turns. Including a safety margin of 1 m to each side, this results in a minimum required room size of 17x17 m for continuous circular wing-borne flight, which is about three times less than for a unidirectional transition. If the available room size is smaller than that, the maximum allowed velocity must be reduced accordingly, resulting in a combination of thrust- and wing-borne flight.

2.3 Indoor position and heading determination

The main application for *Idefix* is autonomous indoor flight. Flying inside enclosed rooms leads to different prob-

lems for the navigation solution than flying outdoors. While GNSS systems are usually used to determine the position outside buildings, they are often unavailable indoors. Also the orientation along the earth's magnetic field is also only possible to a limited extend, since the magnetic field can be strongly disturbed by building structures or electric components. To provide a reliable navigation solution, alternative approaches can be used, for example camera-based [4], ultra-wide-band (UWB) [5] or ultrasound based positioning systems (USP) [6]. The applicability of these systems mostly depends on the aircraft and the accompanied requirements such as accuracy, update rate, weight, range, latency and size.

For the tracking of *Idefix* we employ the ultrasound-based positioning system *Marvelmind* [6]. This multilateration system estimates the aircraft's position by measuring the run time of the ultrasound signal using stationary and mobile beacons, see Figure 5.

Figure 5: Architecture of the USP-System *Marvelmind*

By attaching two mobile beacons with known spacing to *Idefix*, the aircraft's heading can also be estimated from the beacon's relative position vector. The system specifications of *Marvelmind* are listed in Table 2.3. Using an extended Kalman filter [7], the position measurements are fused with the onboard inertial measurement unit (IMU) data to improve the position and heading estimation. The estimation uncertainties resulting from the measurement behavior of the USP are also considered in the simulation.

mass mobile beacon	6.5 g
size mobile beacon	30 x 30 x 20 mm
position deviation	±20 mm
update rate	8 Hz
latency	0.125 s
range	30 m

Table 1: Specifications of the USP system “Marvelmind” [6]

3 CONTROLLER IMPLEMENTATION

Idefix's flight controller runs with a sample time of $T = 0.05$ s and is divided into following cascades: guidance, translational and attitude controller, see Figure 6. In the following section, these cascades are presented with focus on the translational controller.

3.1 Guidance Controller

Since the translational cascade only controls translational accelerations, a super-ordinate guidance controller is necessary. For testing and validation purposes, we use a horizontally and vertically saturated proportional derivative speed controller, enabling straight flight in horizontal x_h -, y_h - and z_h -direction. Control deviations on velocity level are projected onto the commanded accelerations $\vec{a}_c$. Gains for the guidance controller were determined iteratively by simulations to allow for minimum overshoot.

3.2 Translational Controller

The aircraft's capability to transition from vertical thrust-borne flight to horizontal wing-borne flight is accompanied by a complex non-linear characteristics of the translational motion. Extensive knowledge of the flight system's aerodynamics in all flight states is typically required to control this motion. However, small tilting aircrafts, such as *Idefix*, operate in a low Reynolds number regime of $0 < Re < 70000$. Element-based models, which are used for conventional controller design, come with significant uncertainties [8, 9]. Therefore, a translational controller based on the concept of INDI is applied for *Idefix*, allowing for robust control with lower requirements on the aircraft model accuracy [10, 11].

When INDI is applied to the translational motion of *Idefix*, the roll angle, pitch angle, specific thrust $t_m = T_m/m$ and tilt angle of the wing are identified as control variable vector $\vec{u} = [\phi, \theta, t_m, \sigma]^T$. The corresponding control law is given in Equation 6, which calculates the control variable vector u based on the control difference between the commanded acceleration $\vec{a}_c$ and the aircrafts translational acceleration $\vec{a}$ [11].

$$\vec{u} = \vec{u}_f + B^{-1}(\vec{a}_c - \vec{a}) \quad (6)$$

Equation 6 can also be reformulated in incremental form, see Equation 7. This reformulation yields the change in the control variables, depending on the inverse of the control effectivity matrix B and the commanded deviation from the controlled state.

$$\Delta \vec{u} = \vec{u} - \vec{u}_f = B^{-1}(\vec{a}_c - \vec{a}) \quad (7)$$

The control effectivity matrix B is derived from a systems linearization and describes the change in acceleration caused by a change in control variables. This matrix strongly depends on the current state of the control variables, leading to complex interrelations between them. In contrast to the rotational INDI, translational accelerations a are measured directly and do not need to be derived from velocity

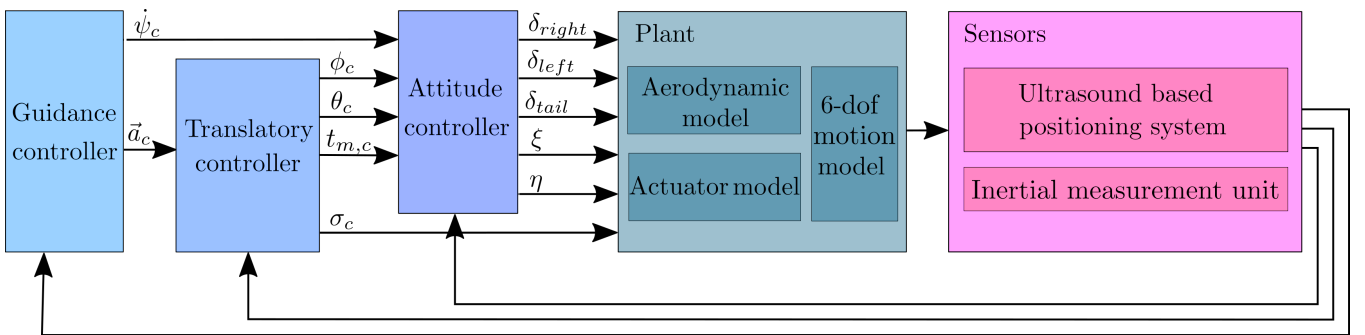


Figure 6: Controller Architecture

measurements. This is also true for the feedback of the control variables ϕ and θ which are direct outputs from the navigation solution. However the control variables t_m and σ are yet to be estimated by dynamic models analogous to the rotational INDI [12].

Coordinate Systems

Choosing coordinate systems and rotation orders of the attitude angles that do not show any singularities in the attitude's description within the entire flight envelope is necessary to describe the control effectiveness. Following, the required coordinate systems are presented.

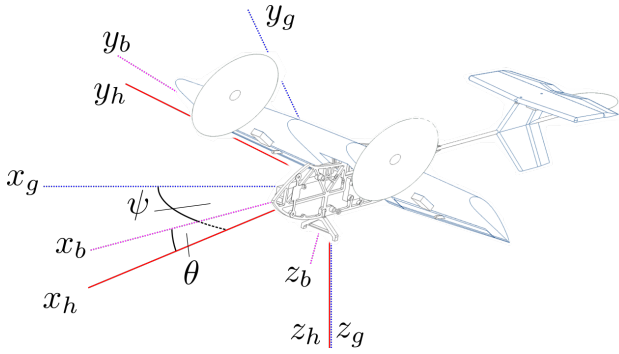


Figure 7: Geodetic, horizontal and body-fixed coordinate system

Figure 7 illustrates the horizontal coordinate system indexed $\square_h$ along with the body-fixed coordinate system indexed $\square_b$ and geodetic coordinate system indexed $\square_g$. In the horizontal coordinate system x_h corresponds to the projection of the body fixed axis x_b onto the x_g - y_g -plane. The z_h -axis is equivalent to the z_g -axis and the y_h -axis lies in the horizontal plane with its direction resulting from the convention of a right turning coordinate system. This allows for formulating the aircraft's attitude with respect to the horizontal frame according the standard rotation order ψ, θ, ϕ with $\psi = 0$.

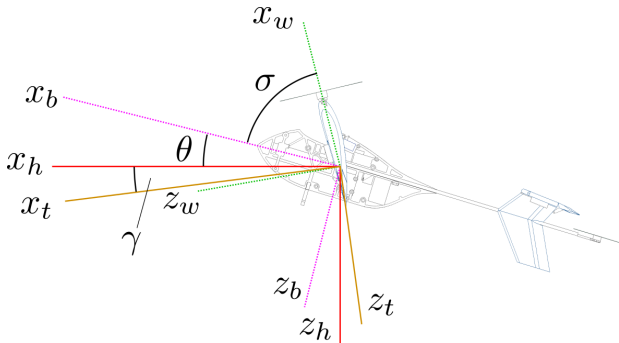


Figure 8: Trajectory, horizontal, body- and wing-fixed coordinate system

The wing-fixed coordinate system indexed $\square_w$ is shown in Figure 8. This coordinate system represents the rotation of the body-fixed coordinate system around the y_b -axis about the wing tilt angle σ [12]. Figure 8 also shows the trajectory coordinate system indexed $\square_t$, which is rotated about the slip angle β_f and the slope angle γ with respect to the horizontal coordinate system.

For the translational INDI a reference frame in horizontal coordinates is chosen. All changes in direction and magnitude of the thrust $\vec{T}$, aerodynamic load $\vec{A}$ and weight force $\vec{G}$ due to changes in the control variables are summarized in the control effectivity matrix B . However, the weight force vector remains unaffected in this reference frame, yielding the zero-entry-matrix $B_{G,h}$. The thrust- and aerodynamic force-related effectivities $B_{T,h}$ and $B_{G,h}$ are more complex and described below.

Thrust control effectivity

Since the tail motor produces less than 10 % of the main motor thrust and is primarily designated to maintain the pitch equilibrium in thrust-borne and slow forward flight, only the main motor effectivity is considered in the translational control effectivity model. The main motors are aligned with the wing chord and the thrust force can be written as follows:

$$\vec{T}_w = \begin{bmatrix} T_m \\ 0 \\ 0 \end{bmatrix}_w \quad (8)$$

Using a thrust model, the thrust force T_m can be estimated from the current throttle position δ_T and the estimated incident flow velocity. The acceleration in the horizontal frame resulting from this thrust force depends on the aircraft mass m and the current transformation relation $M_{hw} = f(\phi, \theta, \sigma)$.

$$\vec{a}_{T,h} = \frac{1}{m} M_{hw} \vec{T}_w \quad (9)$$

Differentiation of Equation 9 for the control vector $\vec{u} = [\phi, \theta, t_m, \sigma]^T$, allows to derive the corresponding control effectivity $B_{T,h} \in \mathbb{R}^{3 \times 4}$:

$$B_{T,h} = \frac{d\vec{a}_{T,h}}{d\vec{u}} = \frac{1}{m} \left(\frac{dM_{hw}}{d\vec{u}} \vec{T}_w + M_{hw} \frac{d\vec{T}_w}{d\vec{u}} \right) \quad (10)$$

Appendix A.1 shows the complete derived thrust related effectivity matrix. Due to the transformation relations from the wing coordinate system into horizontal coordinate system, dependencies of the angles ϕ, θ, σ are included in the formulation of $B_{T,h}$. The change of the thrust force magnitude is only dependent on the change of throttle position, and thus the expression $d\vec{T}_w/d\vec{u}$ can be written as $[0, 0, dT_m/d\delta_{T_m}, 0]^T$.

Aerodynamic control effectivity

The determination of the acting aerodynamic forces via an aerodynamic model is challenging for *Idefix* because of its highly Reynolds number-dependent lift and drag polars at angles of attack in a range of $-180^\circ < \alpha < 180^\circ$ [9]. Even though models for airfoil lift and drag extrapolation exist in literature [1], they show noticeable uncertainties, especially for $|\alpha| \gg 0^\circ$. Alternatively, the aerodynamic forces A_h can be obtained from the force equilibrium in horizontal coordinates according to Equation 11, with $\vec{G}_h$ as the gravitational force, $\vec{T}_h$ as the motor thrust and $\vec{a}_h$ as the aircraft's measured acceleration.

$$\vec{A}_h = \vec{a}_h \cdot m - \vec{G}_h - \vec{T}_h \quad (11)$$

The aerodynamic forces are analyzed in the aerodynamic coordinate system, where they can be divided into lift and drag forces. The aerodynamic force in y_h -direction is assumed to be negligible since only small slip angles are anticipated during both transition and wing-borne flight. Thus, after transforming into the aerodynamic coordinate system with use of the transformation matrix M_{ah} , the aerodynamic force can be written as follows:

$$\vec{A}_a = M_{ah} \vec{A}_h \approx \begin{bmatrix} D \\ 0 \\ L \end{bmatrix}_a \quad (12)$$

The resulting acceleration and the related control effectivity in horizontal coordinates can be described with Equation 13 and 14 with M_{ha} being the transformation matrix from aerodynamic to horizontal coordinate system.

$$\vec{a}_{A,h} = \frac{1}{m} M_{ha} \vec{A}_a \quad (13)$$

$$B_{A,h} = \frac{d\vec{a}_{A,h}}{d\vec{u}} = \frac{1}{m} \left(\frac{dM_{ha}}{d\vec{u}} \vec{A}_a + M_{ha} \frac{d\vec{A}_a}{d\vec{u}} \right) \quad (14)$$

At this point a timescale separation-based assumption affecting the derivatives of the aerodynamic force has to be made. It is assumed, that the aircraft's attitude changes significantly faster than its trajectory. The slope angle γ and the slip angle $\beta_f = \chi - \psi$ of *Idefix* therefore remain constant between two controller time steps. Thus, the transformation matrix M_{ha} can be written as a function of $\phi, \theta, \sigma, \gamma$ and β_f . This also means that the change in the wing's angle of attack α_w is directly influenced by a change in the translational control variables resulting in an analytical description, shown in Appendix A.3. Consequently, Equation 14 can be transformed into Equation 15.

$$B_{A,h} = \frac{d\vec{a}_{A,h}}{d\vec{u}} = \frac{1}{m} \left(\frac{dM_{ha}}{d\vec{u}} \vec{F}_a + M_{ha} \frac{d\vec{F}_a}{d\vec{\alpha}_w} \frac{d\vec{\alpha}_w}{d\vec{u}} \right) \quad (15)$$

Assuming that only the wing induces the aerodynamic force, the derivative $d\vec{F}_a/d\vec{\alpha}_w$ can be expressed with Equation 16 and 17.

$$\frac{\delta L}{\delta \alpha_w} = \frac{\rho}{2} \cdot v^2 \cdot S \cdot C_{L,\alpha_w} \quad (16)$$

$$\frac{\delta D}{\delta \alpha_w} = \frac{\rho}{2} \cdot v^2 \cdot S \cdot C_{D,\alpha_w} \quad (17)$$

These equations include knowledge of the derivatives of the lift and drag polars with respect to the angle of attack C_{L,α_w} and C_{D,α_w} . Without conducting wind tunnel experiments, these derivatives can only be obtained by differentiating the corresponding polars gained from the Montgomery approximation with respect to the angle of attack. Since the extrapolated polars of C_L and C_D are subject to uncertainties, so too are the derivatives C_{L,α_w} and C_{D,α_w} . However, the absolute changes in lift and drag $dL/d\alpha_w$ and $dD/d\alpha_w$ are quadratically proportional to the airspeed and significant control effectivities are only achieved at higher airspeeds, which are accompanied with smaller angles of attack and thus smaller errors in the polars.

To describe the change in the aerodynamic forces between two controller time steps more precisely, their dependency on a change in airspeed also needs to be considered. Since the airspeed is no direct function of the control variables, the derivative of the aerodynamic force with respect to the control variables can not be described. However, the inflow velocity is assumed to change slowly between two controller time steps and the induced change in aerodynamic force is considered negligible in context of the timescale separation. Taking all described aerodynamic effects into account, the aerodynamic related effectivity matrix $B_{A,h}$ is yielded, see A.2. The entire control effectivity matrix B can then be summed up according to Equation 18.

$$B = B_{G,h} + B_{T,h} + B_{A,h} \quad (18)$$

3.3 Attitude Controller

The attitude controller cascade, see Figure 6, is based on a rotational INDI controller described in [11, 13, 14], which determines the control inputs of the aircraft's main motors $\delta_{right}, \delta_{left}$, tail motor δ_{tail} , ailerons ξ and elevator η based on the control deviation in angular acceleration. The transfer function of the rotational INDI is approximated with the dynamic model of the aircraft's control devices. This allows for the use of a linear quadratic regulator (LQR) to calculate the optimal commanded angular accelerations based on the control deviation in thrust and attitude angles. Due to the coupling of attitude angles and the translational controller described in 3.2, the attitude control behaviour, analysed in [13], is included in the interpretation of the simulation results.

4 SIMULATION

The simulation is implemented in MATLAB/Simulink according to the dynamic model described in Section 2.1. The response to commanded velocity steps of ± 1 m/s in all translational axes is analyzed in both thrust- and wing-borne flight to verify the controller. The control performance in a velocity range of 0 – 8 m/s is also examined to account for the aircraft's ability to transition from thrust to wing-borne flight. In all cases, the translatory accelerations are limited to ± 1 m/s², and all velocity steps are applied separately in all axes. The heading of the aircraft is kept constant for all simulations.

4.1 Thrust-borne flight

Figure 9 shows the step response of the velocity controller in thrust-borne flight. A similar response behavior can be observed in all axes, with a rise time of about 1.3 s for both acceleration and deceleration with no noticeable overshoot. The response is slightly delayed at the beginning of the steps, resulting from *Idefix*'s mass inertia and actuator dynamic.

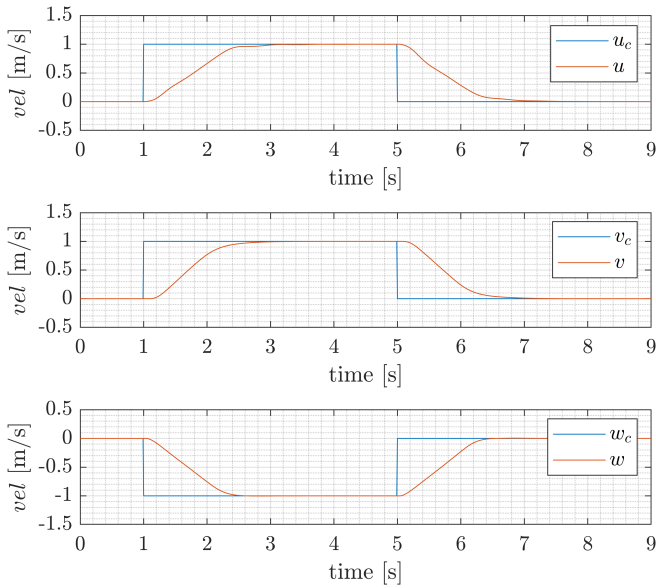


Figure 9: Velocity steps in thrust-borne flight (0 m/s)

This behavior can be observed more clearly in the response behavior of the corresponding accelerations in Figure 10. It can also be seen, that the delay in y_h -direction is about 0.1 s more than in x_h - and z_h -direction. Since an acceleration in y_h -direction can only be indirectly achieved by a change in roll angle, this dynamic is significantly slower than accelerations in x_h - and z_h -direction, which are directly controlled by the control devices' main motor thrust and tilt angle. It is also noticeable, that the commanded maximum acceleration is not reached in the time-span of the maneuver. This can be explained by the change of drag force induced

by the change in velocity, which was neglected in the controller design due to the argumentation in Section 3. However, neglecting this effect has only a minor impact on the acceleration response behavior and does not significantly affect the control performance.

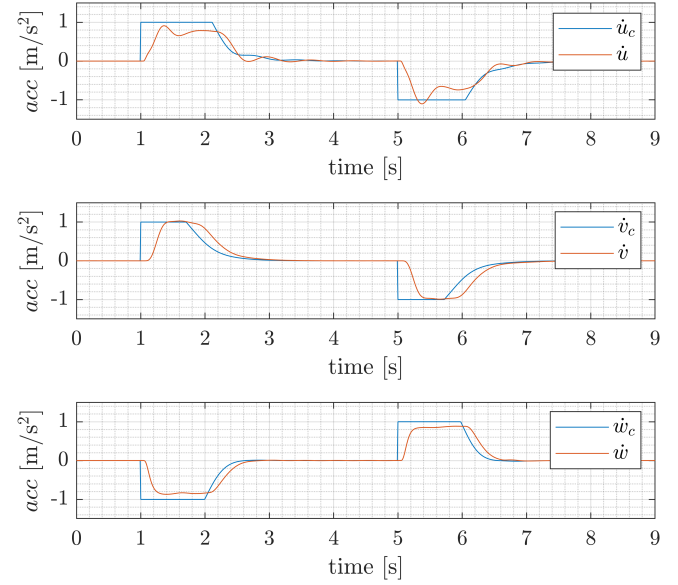


Figure 10: Accelerations in thrust-borne flight (0 m/s)

4.2 Wing-borne flight

In wing-borne flight, the velocity controller's step response shows similar behavior to that of the thrust-borne flight, see Figure 11.

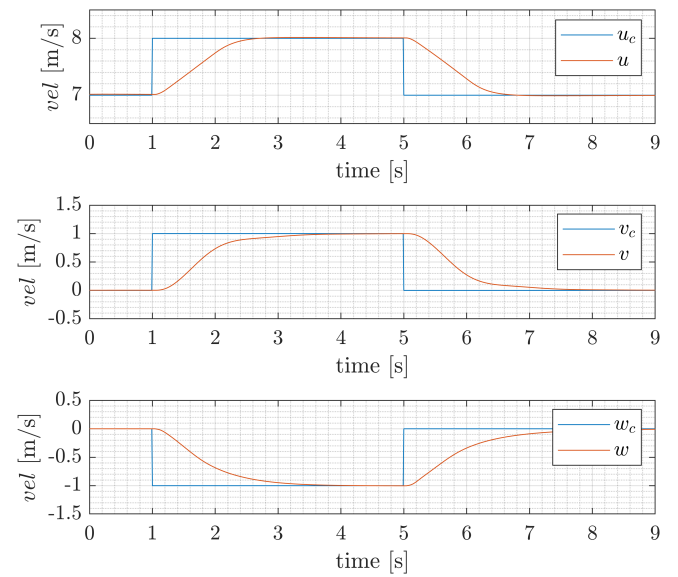


Figure 11: Velocity steps in wing-borne flight (7 m/s)

The only noticeable difference is a slightly longer rise time of about 1.5 – 2 s. This effect can consequently be observed in the corresponding acceleration response, see Figure 12. The change in dynamic can be explained by the resistance increasing quadratically with airspeed and the change of the control variable responsible for the movement in y_h - and z_h -direction.

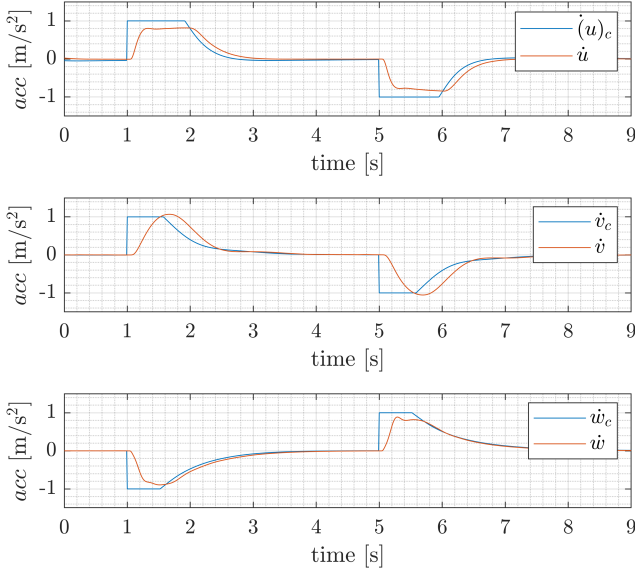


Figure 12: Acceleration in wing-borne flight (7 m/s)

Whereas in thrust-borne flight, the main motor was directly responsible for acceleration in z_h -direction, in wing-borne flight this is taken over by the change in pitch angle, which primarily changes the aircraft's attitude and subsequently the vertical acceleration. In y_h -direction, the acceleration is still induced by a change in roll angle, but the control device on attitude controller level changes from differential thrust to aileron deflection [11]. Even though these control devices have a similar dynamic, the roll damping in wing-borne flight is quadratically proportional to the airspeed [15] and therefore much higher than in thrust-borne flight. Since the control performance on velocity controller level is still sufficient, the assumptions made in the attitude controller and Chapter 3 can be considered valid.

4.3 Transition from thrust- to wing-borne flight

The control performance of the controller during the transition from thrust- to wing-borne flight is shown in Figure 13. During the transition, a steady increase in velocity at about 1 m/s can be observed with no noticeable influence on the velocities in y_h - and z_h -direction. This confirms the applicability of the translational INDI controller to the entire velocity range of the highly unsteady transition from thrust- to wing-borne flight.

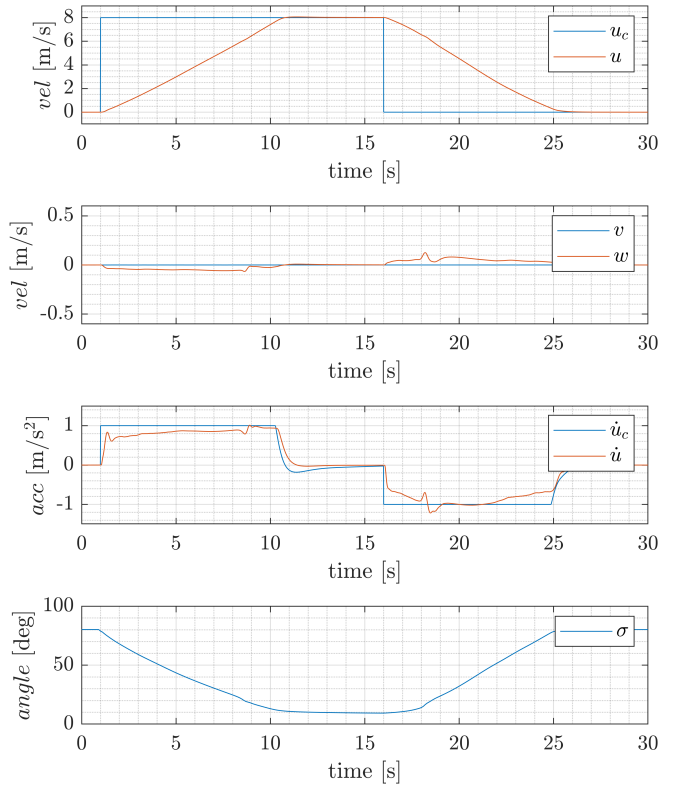


Figure 13: Transition from 0 – 8 m/s

5 CONCLUSION

In this paper, a concept for operating a miniaturized tilting aircraft in confined spaces was presented. Taking into account tilting-specific flight dynamic constraints and a maximum longitudinal acceleration of $\pm 1 \text{ m/s}^2$, it was shown, that an operation area $17 \times 17 \text{ m}$ is sufficient for a full transition from thrust- to wing-borne flight along a circular path. Due to the absence of GNSS signals within enclosed spaces, an alternative concept based on ultrasound was employed to determine the aircraft's position and heading with an accuracy of $\pm 0.02 \text{ m}$ and $\pm 2^\circ$.

For the aircraft's translational control, a cascaded INDI based approach was implemented, showing complex interrelations between control variables and control effectivities. To improve the model of the control effectivities, in-flight measured aerodynamic forces were used in addition to standard lift and drag models. The functionality of the controller was validated in simulations with focus on the translational cascade. On velocity and acceleration level a stable and agile response behavior was achieved for thrust-born and wing-borne flight as well as the transition between both.

Future work will focus on translating this simulative work into fully autonomous real flights. Along with the controller

implementation of a circular transition pattern at a fixed height, this also includes a validation of the selected indoor navigation system and the integration in the existing navigation solution.

REFERENCES

- [1] B. Montgomerie. Methods of Root Effects, Tip Effects and Extending the Angle of Attack Range to $\pm 180^\circ$, with Application to Aerodynamics for Blades on Wind Turbines and Propellers. *Swedish Defence Research Agency*, 2004.
- [2] P. Hartmann. Vorausschauende Flugbahnregelung fuer Kippflugelflugzeuge. 2017.
- [3] I. Barz. Flugbahnplanung für Kippflügelflugzeuge zur Unterstützung im Rettungseinsatz. 2023.
- [4] OptiTrack. <https://optitrack.com/>. Accessed: 2023-05-01.
- [5] A. Masiero, F. Fissore, R. Antonello, A. Cenedese, and A. Vettore. A comparison of UWB and motion capture uav indoor positioning. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 42(2/W13):1695–1699, 2019.
- [6] Marvelmind Robotics. Ultrasonic positioning system <https://marvelmind.com/>. Accessed: 2023-05-01.
- [7] J. Wendel. *Integrierte Navigationssysteme: Sensor-datenfusion, GPS und Inertiale Navigation*. De Gruyter, 2011.
- [8] Justin Winslow, Hikaru Otsuka, Bharath Govindarajan, and Inderjit Chopra. Basic understanding of airfoil characteristics at low reynolds numbers (10 4–10 5). *Journal of aircraft*, 55(3):1050–1061, 2018.
- [9] Airfoil Tools. <http://airfoiltools.com/airfoil/details?airfoil=naca4415-il#polars>. Accessed: 2023-05-01.
- [10] E. 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, 2018.
- [11] Fabian Binz. Robust, Fault-Tolerant Control of Aircraft with Hovering Capability, 2020. Aachen.
- [12] 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.
- [13] J. Müller D. Moormann. Miniaturization and Control of an Unmanned Tiltwing Aircraft. In *IEEE International Micro air vehicle conference and competition (IMAV)*, 2022.
- [14] D. Schatten D. Moormann. Attitude control for a tiltwing aircraft under tail actuator failures. In *IEEE International Micro air vehicle conference and competition (IMAV)*, 2022.
- [15] 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.

APPENDIX A: DATA

A.1 Thrust control effectivity

$$B_{T,h} = \begin{bmatrix} t_m \lambda_1 & -t_m \lambda_7 & -t \lambda_8 & \frac{dt_m}{d\delta_t} \lambda_6 \\ t_m \lambda_2 & 0 & t \lambda_4 & \frac{dt_m}{d\delta_t} \lambda_5 \\ t_m \lambda_3 & -t_m \lambda_6 & t \lambda_9 & -\frac{dt_m}{d\delta_t} \lambda_7 \end{bmatrix} \quad (19)$$

columns: $\phi, \theta, \sigma, t_m$
rows: $\dot{u}, \dot{v}, \dot{w}$

where:

$$\begin{aligned} \lambda_1 &= \sin(\phi) \sin(\sigma) \sin(\theta) \\ \lambda_2 &= \cos(\phi) \sin(\sigma) \\ \lambda_3 &= \cos(\phi) \sin(\sigma) \sin(\theta) \\ \lambda_4 &= \cos(\sigma) \sin(\phi) \\ \lambda_5 &= \sin(\phi) \sin(\sigma) \\ \lambda_6 &= \cos(\theta) \cos(\sigma) - \cos(\phi) \sin(\theta) \sin(\sigma) \\ \lambda_7 &= \sin(\theta) \cos(\sigma) + \cos(\phi) \cos(\theta) \sin(\sigma) \\ \lambda_8 &= \cos(\theta) \sin(\sigma) + \cos(\phi) \sin(\theta) \cos(\sigma) \\ \lambda_9 &= \sin(\theta) \sin(\sigma) - \cos(\phi) \cos(\theta) \cos(\sigma) \end{aligned}$$

A.2 Aerodynamic control effectivity

$$B_{A,h} = \begin{bmatrix} \left[\frac{d\tilde{a}_{A,h}}{d\phi} \right]_x & \left[\frac{d\tilde{a}_{A,h}}{d\theta} \right]_x & \left[\frac{d\tilde{a}_{A,h}}{d\sigma} \right]_x & 0 \\ \left[\frac{d\tilde{a}_{A,h}}{d\phi} \right]_y & \left[\frac{d\tilde{a}_{A,h}}{d\theta} \right]_y & \left[\frac{d\tilde{a}_{A,h}}{d\sigma} \right]_y & 0 \\ \left[\frac{d\tilde{a}_{A,h}}{d\phi} \right]_z & \left[\frac{d\tilde{a}_{A,h}}{d\theta} \right]_z & \left[\frac{d\tilde{a}_{A,h}}{d\sigma} \right]_z & 0 \end{bmatrix} \quad (20)$$

columns: $\phi, \theta, \sigma, t_m$
rows: $\dot{u}, \dot{v}, \dot{w}$

A.3 Angle of attack of the wing in horizontal coordinates

$$\alpha_{w,h} = \tan^{-1} \left(\frac{\sin(\gamma)\epsilon_1 + \epsilon_2\epsilon_3 - \epsilon_4}{\sin(\gamma)\epsilon_5 + \epsilon_2\epsilon_6 + \epsilon_7} \right) \quad (21)$$

where:

$$\begin{aligned} \epsilon_1 &= \sin(\sigma) \sin(\theta) - \cos(\phi) \cos(\sigma) \cos(\theta) \\ \epsilon_2 &= \cos(\beta_f) \cos(\gamma) \\ \epsilon_3 &= \cos(\theta) \sin(\sigma) + \cos(\phi) \cos(\sigma) \sin(\theta) \\ \epsilon_4 &= \cos(\gamma) \sin(\beta_f) \cos(\sigma) \sin(\phi) \\ \epsilon_5 &= \cos(\sigma) \sin(\theta) + \cos(\phi) \cos(\theta) \sin(\sigma) \\ \epsilon_6 &= \cos(\sigma) \cos(\theta) - \cos(\phi) \sin(\sigma) \sin(\theta) \\ \epsilon_7 &= \cos(\gamma) \sin(\beta_f) \sin(\phi) \sin(\sigma) \end{aligned} \quad (22)$$

Performance Comparisons on Fully-Actuated Multirotors with Different Actuation Modes

Caleb Probine, and Karl A. Stol*

University of Auckland, Department of Mechanical and Mechatronics Engineering, Auckland 1010, New Zealand

ABSTRACT

For the interaction of Unmanned Aerial Vehicles (UAVs) with the environment, their ability to station-keep is critical. In conventional station keeping, a UAV will change attitude to translate, and reject disturbances. Meanwhile, a fully-actuated UAV can exert these forces without rolling or pitching: this is also known as vectored thrust. These two methods of position control represent two actuation modes, with very different dynamics. The work presented in this paper investigates how Proportional-Integral-Derivative (PID) controllers can be tuned to compare the two actuation modes discussed in wind disturbance rejection tasks. The comparison is made fair by ensuring that actuator usage is the same between the two cases. Overall, when tuned for moderate control precision, it is found that the actuator usage is significantly higher when vectored thrust is used, making it less favourable for general station keeping tasks. However, linear model studies indicate that higher precision may necessitate the use of vectored thrust, rather than conventional methods.

1 INTRODUCTION

Given their numerous possible applications, such as agricultural assessments [1], traffic estimation [2], and power line inspection [3], UAVs have been increasing in their civil applications in recent years [4]. Within the range of different craft configurations, multirotors are particularly favourable for their ability to hover in place. Another unique capability of multirotors is their ability to interact with the environment, an example of which is given in [5]. In these environmental interaction tasks, a range of control issues exist such as the requirement to handle contact forces, and reject wind disturbances effectively.

Within the realm of multirotors, the class of over-actuated/fully-actuated crafts are of particular interest in these applications. These have the property of being able to generate a full 6-DoF wrench vector, and as a result, can regulate position and orientation independently. This is in contrast to

under-actuated crafts, which must roll or pitch to translate in the horizontal plane: the more conventional form of actuation. This wrench generation capability is a critical craft-capability for the handling of tools in the environment without external manipulators. Two common methods for achieving this full-actuation include the introduction of tilting rotors such as in [6], or the use of fixed-tilt rotors, such as in [7].

Whilst fully-actuated crafts must be used to allow full-pose tracking, there are various trade-offs when compared to under-actuated crafts. One of the core disadvantages of fully-actuated crafts are their lower efficiency, and this is documented well in a range of works. For example, in [8], the wastage of internal forces in fully-actuated crafts is articulated with a force efficiency index. A more dynamic observation of the inefficiency can be observed in [9]. In this work, for a fully-actuated craft, the frequency response of motor speeds to translation commands is compared for conventional actuation, and vectored thrust (translating without attitude-change). The latter, fully-actuated method of translation is found to result in higher rotor speeds at almost all frequencies. In [10], a craft that can transform between co-planar (under-actuated) and omni-directional modes is presented. Via a common mission, it is then shown that the energy usage when undertaking pure fully-actuated omni-directional flight is higher than that when the craft is converted between co-planar and fully-actuated modes as appropriate. However, a critical advantage in fully-actuated crafts is the faster response to forces. This is documented in the context of disturbance rejection in [11], and is experimentally verified.

Throughout works that present new crafts and control methods, the performance under the unique pose-tracking capability is frequently demonstrated: for example the work in [12] clearly showcases the omni-directionality of the new craft. However it is also of interest to benchmark the performance of fully-actuated UAVs against their under-actuated counterparts in common tasks. Some works that have addressed this are described below.

In [13], a tilting-rotors craft is described, and made to follow a trajectory under a wind disturbance. This experiment is undertaken in both 'conventional' and 'tilt-augmented' configurations, with the former being generated by locking the rotor tilting mechanism. Under identical control gains, poorer performance (higher overshoot) is observed with the tilt-augmented configuration. This method of making comparisons by changing the actuation modes used by the craft is far better than the alternative of comparing different crafts. A

*Email address: k.stol@auckland.ac.nz

similar approach has also been undertaken in [6, 14, 15].

In the comparisons presented in [6], mean absolute and standard deviation of position are used as comparison metrics in a common step tracking task, and similar performance is observed between the platforms. However, the comparison's validity is limited by vagueness in the controller tuning processes. Meanwhile, the authors of [14] tune controllers for each craft configuration with similar step responses, following which, their (sinusoidal) disturbance rejection performance is compared, showing an advantage for fully-actuated crafts. Finally, the authors of [15] compare tethered conventional and tilting rotors quad-copters under atmospheric wind disturbances. In this case, root mean square (RMS) position error is used to quantify position, with an advantage demonstrated for the tilting-rotors craft. Whilst these works present a range of useful metrics for assessing position tracking performance when comparing under-actuated and fully-actuated crafts, disparate conclusions are reached about their relative performance. This is due to inconsistent, rigor-limited tuning of the controllers.

Work that does approach closed-loop comparisons appropriately is presented in [16]. This work takes the approach of using non-linear dynamic inversion, to ensure that the position response between 3 multirotor configurations, is identical. Within these configurations both under-actuated and fully-actuated craft are included. In simple polynomial trajectory tracking simulations the actuator usage of these crafts is then compared, and the fully-actuated craft is shown to require higher actuator inputs to follow the trajectory. This consideration of position tracking and actuator usage makes for a highly rigorous comparison, however it is limited by the simplicity of the model used, which don't reflect aerodynamic effects, or motor models. It is also limited by the simplicity of the trajectory used.

In comparing actuator usage a range of metrics exist. In the context of optimising attitude controllers, the weighted $\mathcal{H}_\infty$ norm of the actuator usage is constrained in [17]. Meanwhile, in position control optimisation for wind-rejection in [9], the $\mathcal{H}_\infty$ norm of control input is added to the weighting when designing a controller. In simulation, different metrics are then used again. For example, in [18], Integral absolute input is used as a measure of force variation. Meanwhile, [9] analyses standard deviation in each motor's pulse-width-modulation (PWM) when assessing actuator usage.

In terms of the control techniques used on fully-actuated crafts, a range of advanced control schemes have been implemented, including sliding mode control [19], model predictive control [20] and adaptive control [21] among others. However, in [6, 15, 13], also observed was the use of simple PID and PD components within the control systems. Overall, PID controllers are useful for their simplicity and robustness, and they provide a good starting point for analysing craft control performance. A number of additional works have used simple controllers, often related to PID, with examples in-

cluding [10, 22, 23]. In some cases, lower level loops are augmented by feedback-linearised or geometric components, such as in [12].

Overall, this work aims to contribute to more rigorous comparisons of fully-actuated crafts in the context of disturbance rejection. In particular, PID tuning will be used as a basis to investigate the relative efficiency on a craft that is tuned to have similar performance in its fully-actuated and under-actuated modes.

The remainder of this paper is structured as follows. Section 2 gives a brief outline of the craft used in the comparisons undertaken. Section 3 then details the model of said craft is constructed. In addition, the methodologies for fair tuning comparisons are also presented. Section 4 then provides the results of these studies, after which Section 5 provides analyses of how the results change when the aggressivity of control changes.

2 MULTIROTOR DESCRIPTION AND ACTUATION MODES

The craft used as a basis for this comparison is shown in Figure 1. It has a flying mass of 1.72 kg and a rotor-to-rotor diameter of 500 mm . It was constructed, and modelled in [9], based on optimisation algorithms outlined in [24]. This craft is technically over-actuated, as it has 2 more motors than required for full actuation, however, as the redundancy is not a core point in this work, it shall be referred to it as fully-actuated. When discussing the craft, there are two frames of concern. Vectors in the fixed world frame are denoted with $^{\mathcal{W}}$. Meanwhile, $^{\mathcal{B}}$ is used to denote the body frame, which moves with, and is attached to, the craft.

Like an under-actuated craft, the octorotor can create lateral forces by rolling/pitching, which will be denoted as attitude-based thrust $^{\mathcal{W}}F_A$. However, given its full-actuation, it can also create lateral forces whilst keeping level, which will be denoted vectored thrust $^{\mathcal{W}}F_V$. These two actuation methods are depicted in Figure 2. Throughout the remainder of this paper, the subscript $\cdot_A$ corresponds to attitude-based thrust control, whilst $\cdot_V$ corresponds to vectored thrust control. When controlling position via vectored thrust, the command will always be commanded to stay level, which is



Figure 1: Craft for which control is tuned [9].

impossible with attitude-based thrust. Also note that when attitude-based thrust is utilised, comparisons will still be made using the canted craft: i.e. the craft will remain fully-actuated, but vectored thrust will not be used.

The control software used is the PX4 flight stack, which provides the framework in which control structures are encoded. The outer loops of this control structure are indicated in Figure 3, whilst the inner loops are described in Figure 4. The feedback supplied to the controller in Figure 3 is the position vector $\xi \in \mathbf{R}^3$, which has components as follows: $[\mathcal{W}_x \ \mathcal{W}_y \ \mathcal{W}_z]^\top$. This control must be completed in the presence of wind disturbances ${}^{\mathcal{W}}\mathbf{w}$, which have components in 3 axes. Also of concern is the actuator usage, as encoded by the 8 element vector χ , of the PWM motor commands.

At the highest level, a cascaded controller is used for each axis independently. This cascaded structure has position and speed loops for each axis, with the position loop being P controlled, and the velocity loop being PID controlled. This generates the translational commands for the craft.

For attitude-based thrusts, the translational commands are converted to angle set-points, which are in turn passed to a lower level attitude controller. This is again of a cascade structure, with angle and angle-rate loops respectively. For the angle loop, non-linear P control is used, whilst for the angle-rate loop, PID control is used for each axis. The output of this stage is a set of torques for the craft. Meanwhile, when vectored thrust is used, the desired forces are generated directly. In this case the attitude controller runs in parallel to the position controller, with constant zero references for orientation.

The lowest level component of this is a multirotor mixer: this converts from a vector of force and torque commands to individual motor commands.

3 MODELLING

3.1 Nonlinear Numerical Model

The control assessment methods all use the non-linear octocopter model defined in [9]. This details critical dynamics such as the drone's rigid body motion, a first-order non-linear motor model, alongside an empirical aerodynamic model. These physical control components are then accompanied by components representing the lower layers of the PX4 control architecture. These include a multirotor mixer, an attitude controller, and a conversion from attitude-based thrust commands to angle set-points. Links between these are all shown in Figure 4. This figure also demarcates the inputs and outputs of the plant.

When using attitude-based thrust control, the vector ${}^{\mathcal{W}}F_A$ will be the control input, and ${}^{\mathcal{B}}F_V$ will be zeroed. When using vectored thrust control, the vector ${}^{\mathcal{B}}F_V$ will be manipulated, along with ${}^{\mathcal{W}}F_{A,z}$. Because the craft is level during vectored thrust control ${}^{\mathcal{W}}F_V = {}^{\mathcal{B}}F_V$.

When the loop is closed, the focus is then on the closed-loop plant shown in Figure 3. Note that in this closed-loop

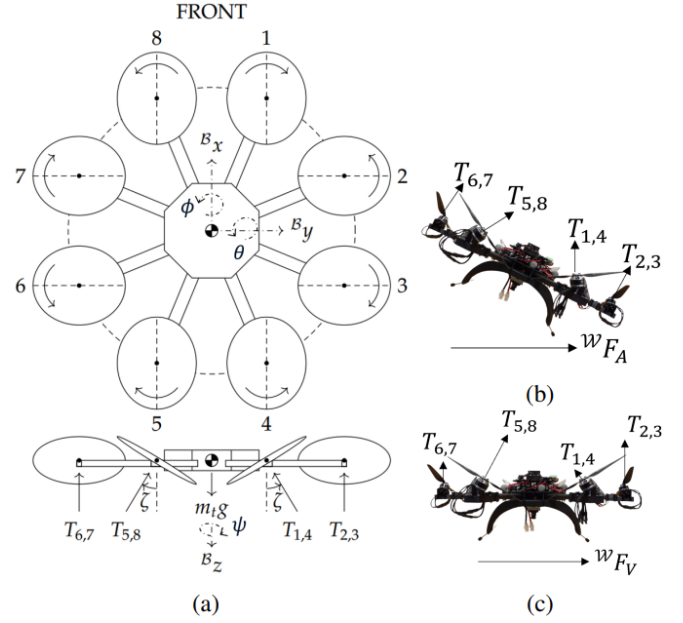


Figure 2: Craft dynamics illustrations. (a) Octocopter Schematic [9]; (b) Motor usage when translating with attitude-based thrust ${}^{\mathcal{W}}F_A$ in the ${}^{\mathcal{W}}y$ direction; (c) Motor usage when translating with vectored thrust ${}^{\mathcal{W}}F_V$ in the ${}^{\mathcal{W}}y$ direction [25]

plant, the reference is zero, and the response of the outputs to a random wind is of interest.

3.2 Linearisation Description

In order to convert the non-linear model to a linear one for analysis, Jacobian linearisation is undertaken. This requires the generation of an operating point about which to linearise, which is found via trim analysis. For trim analysis, the following constraint is applied on the wind

$${}^{\mathcal{W}}\mathbf{w}^* = [5.6 \ 0 \ 0]$$

This wind speed is chosen based on values used in [9]. Depending on the actuation method used, a second constraint is applied. When attitude-based is used,

$$\mathbf{u}^* = [{}^{\mathcal{W}}F_{A,x} \ {}^{\mathcal{W}}F_{A,y} \ {}^{\mathcal{W}}F_{A,z} \ 0 \ 0]$$

is applied. Meanwhile, when vectored thrust is used,

$$\mathbf{u}^* = [0 \ 0 \ {}^{\mathcal{W}}F_{A,z} \ {}^{\mathcal{B}}F_{V,x} \ {}^{\mathcal{B}}F_{V,y}]$$

is applied. It should be noted that in the linearisation process, the attitude control gains are fixed. Also, at this stage, the attitude controller model used is continuous. After MATLAB is used to solve for the entire trim point, the linear model can be generated.

3.3 Controller Tuning Methods

For tuning, existing gains are taken for attitude-based thrust control. These gains correspond to moderate preci-

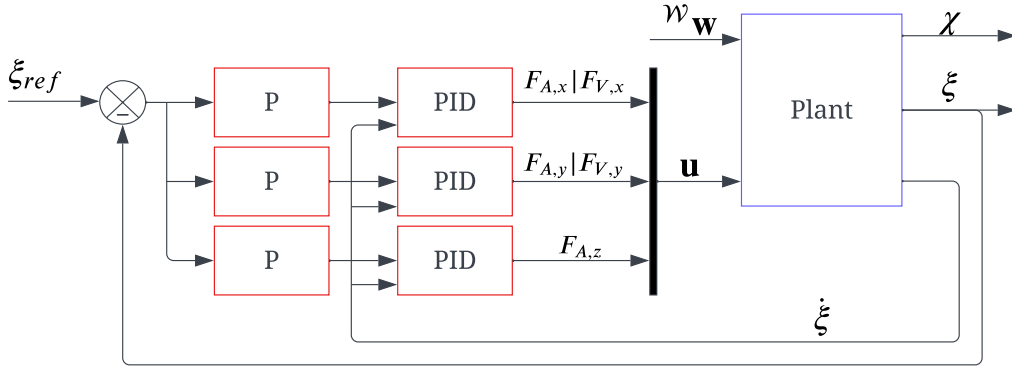


Figure 3: Control structure used in this work. Note that one pair of P-PID controllers exist for each axis. The gains for the first two axes are the same, whilst the ^Wz axis is different. Also note, depending on the actuation method being used, the ^Wx and ^Wy control actions will be mapped to different plant inputs. Finally, note that in the derivative components of the second loop, reference derivatives are not used.

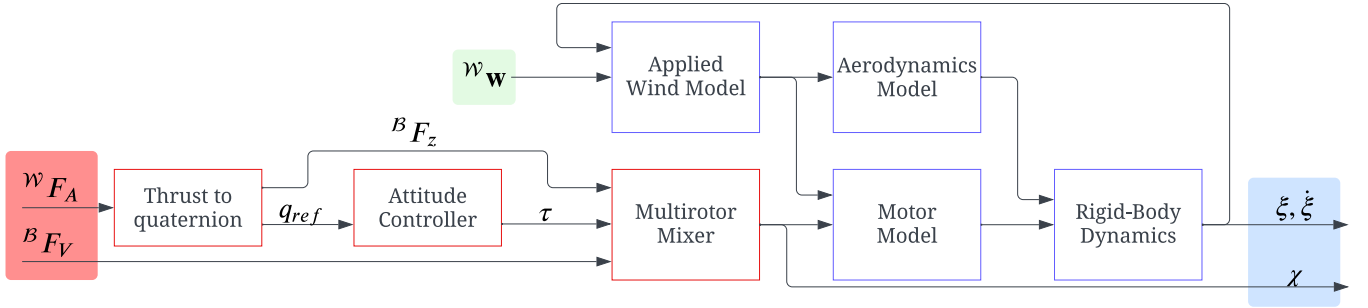


Figure 4: Components comprising the non-linear craft model. Red, green and blue regions respectively denote inputs, disturbance inputs and outputs of the model. Blue-outlined blocks represent physical model elements, whilst red-outlined blocks denote software elements. Note also that attitude feedback amongst these blocks is omitted for clarity.

sion, where there is still significant headroom for the actuators. These are given in Table 1. By closing the loop with attitude-based thrust control, and these gains, a linear model of the closed-loop process is generated. The linear model for attitude-based thrust control is then truncated to focus on the ^Wx axis. From this linear model a transfer function from the ^Wx axis wind, to ^Wx position $D(s) = X(s)/W_x(s)$ can then be generated. For attitude-based thrust control, this is denoted $D_A(s)$.

$K_{p,xy}$	$K_{p,\dot{x}y}$	$K_{i,\dot{x}y}$	$K_{d,\dot{x}y}$
2	0.09	0.02	0.01
$K_{p,z}$	$K_{p,\dot{z}}$	$K_{i,\dot{z}}$	$K_{d,\dot{z}}$
1	0.2	0.1	0
$K_{p,\theta}$	$K_{p,\dot{\theta}}$	$K_{i,\dot{\theta}}$	$K_{d,\dot{\theta}}$
6.5	0.15	0.05	0.003

Table 1: Gains used for attitude-based thrust control. The final row of gains are for the inner attitude loop, and are preserved with vectored thrust control.

The closed-loop plant for vectored thrust control is then chosen such that the corresponding transfer function $D_V(s)$, has the same magnitude as $D_A(s)$, when evaluated at different frequencies. The closed-loop transfer function $D_V(s)$ is manipulated mainly via the P and PID gains for the horizontal axes (xy).

The magnitude equalisation is undertaken for the following reason. With wind represented by unit white-noise, coloured by a transfer function $Q(s)$, the resulting RMS error in position is given by the $\mathcal{H}_2$ norm shown in Equation 1.

$$\|D(s)Q(s)\|_2 = \sqrt{\frac{1}{2\pi} \int_{-\infty}^{\infty} |D(j\omega)Q(j\omega)|^2 d\omega} \quad (1)$$

Hence by ensuring $|D_V(j\omega)| \cong |D_A(j\omega)|$, the position performance between the two actuation methods can be preserved, after which actuation can be compared. This is similar in the approach taken in [16], but applied to a simpler controller, in the context of disturbance rejection.

Using the gains generated, insight is gained into the actuation usage by first looking at the PWM, and its variation with frequency in the linear model. In particular, $p(\omega)$ in Equ-

tion 2 represents the average standard deviation in PWM under a sinusoidal unit wind input, at a given frequency.

$$p(\omega) = \frac{1}{8\sqrt{2}} \sum_{i=1}^8 |V_i(j\omega)| \quad : \quad V_i(s) = \chi_i(s)/W_x(s) \quad (2)$$

To further evaluate the relative performance of the two station-keeping methods, both systems are then simulated with the corresponding gains.

3.4 Simulation Description

For simulation tests of the plant, the non-linear model referenced above was used. However, it was augmented to more accurately represent the real system. This involved replacing the continuous attitude controller, with a discrete time version. Also included was a sensor model with the PX4 Extended Kalman Filter (EKF), and the designed closed-loop controller used for position stabilisation. The wind trajectories used as an input to the system are generated from Turbsim. The profiles used are the same as those used in [9]. The spectrum of the wind for two axes is shown in Figure 5.

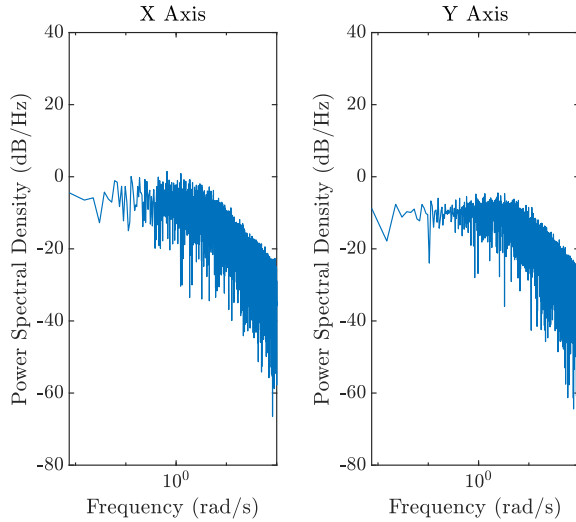


Figure 5: Spectrum of wind speed in the X and Y axes (with mean value removed). Note that the dominant frequencies are below 1Hz. Also, the Z axis is omitted here, but it's spectrum is similar to that in the second.

In order to compare actuation methods in simulation, metrics are generated from measured data. First, RMS position error, from [9], is used and it is given in Equation 3.

$$\sigma_\xi = \sqrt{\frac{1}{M} \sum_{m=1}^M \|\xi_{des,m} - \xi_m\|^2} \quad (3)$$

In Equation 3, $\xi_m \in \mathbf{R}^3$ denotes the position vector each time step m . Meanwhile, $\xi_{des,m}$ denotes the reference position, which is zero at all time steps.

To measure actuator usage, the average motor PWM standard deviation is used, given in Equation 4, from [9].

$$\bar{\sigma}_\chi = \frac{1}{N} \sum_{i=1}^N \sqrt{\frac{1}{M} \sum_{m=1}^M (\chi_{i,m} - \bar{\chi}_i)^2} \quad (4)$$

In Equation 4, χ_i is the PWM of motor i . $\bar{\chi}_i$ denotes the mean PWM of a motor. In both equations M denotes the total number of samples available from simulation.

4 RESULTS

After tuning, the response in Figure 6 is seen, from wind to position, which is clearly matched. The resulting gains for vectored thrust control are shown in Table 2.

$K_{p,xy}$	$K_{p,xy}$	$K_{i,xy}$	$K_{d,xy}$
1.7	0.08	0.02	0
$K_{p,z}$	$K_{p,z}$	$K_{i,z}$	$K_{d,z}$
1	0.2	0.1	0

Table 2: Gains used for vectored thrust control.

The plot of $p(\omega)$ is then shown in Figure 7. It is clearly seen that for most frequencies, the vectored thrust plant will have higher actuator usage (as measured by PWM variation). It is only in the higher frequencies, that vectored thrust has lower actuator usage. This is due to the fact that in this region the disturbances are outside the range of effective frequencies for attitude-based thrust, but still below the bandwidth of vectored thrust. This indicates that despite vectored thrust's superior bandwidth, in most frequencies, it will be less efficient than attitude-based thrust. This further confirms the results from [9] in a closed-loop context.

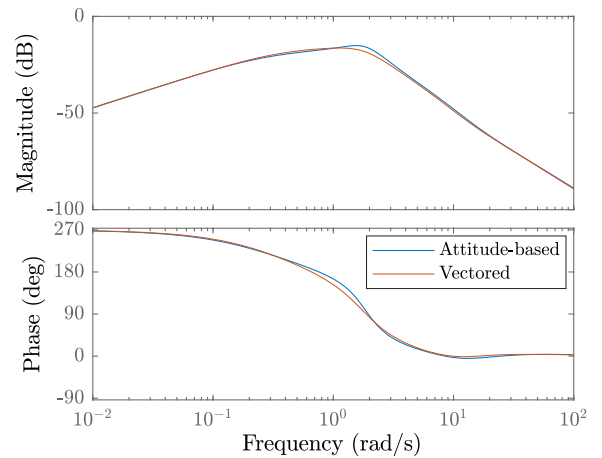


Figure 6: Bode plot from wind to position for the two actuation modes.

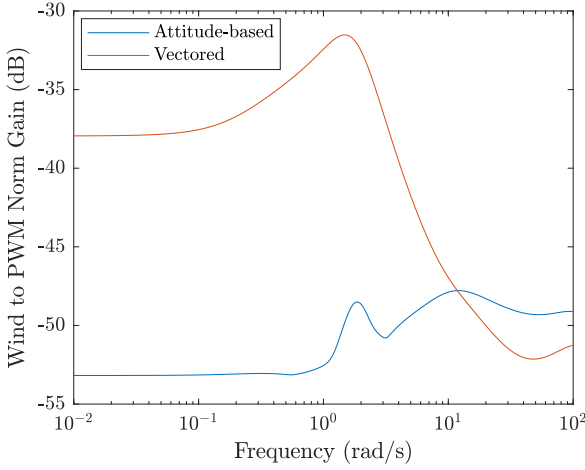


Figure 7: Variation with frequency of the PWM usage, for the two actuation modes.

When these control gains are then simulated, the performance is as described in Table 3.

	$\sigma_{\xi}(mm)$	$\bar{\sigma}_{\chi}$
Attitude-based thrust	66	0.0125
Vectored thrust	66	0.0271

Table 3: Performance parameters for the two tuned plants.

The relative performance of the controllers shown in Table 3 is highly intuitive, as the dominant frequencies in the wind, as seen in Figure 5, mostly overlap with the regions where attitude-based thrust uses much less PWM.

5 DISCUSSION OF CONTROL AGGRESSIVITY

The results in Section 4 correspond to moderate performance. However, also of interest, is how the relative performance of these two actuation methods changes as more aggressive tuning is approached. This is explored here in a linear context via an optimal control problem.

For each actuation method, a linear plant is available. For the linear plant, the position and speed loop controller gains are then optimised to minimise the response of the plant to wind. This is done by minimising the $\mathcal{H}_2$ norm of the transfer function $D(s)$ for each actuation method. This is evaluated under different constraints on PWM, described in Equation 5.

$$\|\chi_i(s)/W_x(s)\|_2 \leq a(\min(1 - \chi_i^*, \chi^*)) \quad \forall i \in \{1, 2, 3, 4\} \quad (5)$$

In Equation 5 the limit on each rotor is scaled according to its distance from saturation at trim. The parameter a then allows the conservativeness to be modified, with higher values corresponding to a more lenient constraint, and a more aggressive controller. In this equation, it should be noted that only the

first 4 motors are used in the constraint. This is because this optimisation focuses on X-axis control, and so symmetry is expected between the motors on each side of the craft.

Control optimisation is undertaken using the 'Systune' tool from MATLAB, which uses optimisation methods discussed in [26]. Ultimately, the optimisation gives results shown in Figure 8, with the a value on the horizontal axis. It can clearly be seen here that for more conservative controllers, the vectored thrust based control performs worse. In this case, under the same actuator usage constraint, vectored thrust has worse position performance. However, under aggressive control, attitude-based thrust control reaches a hard limit.

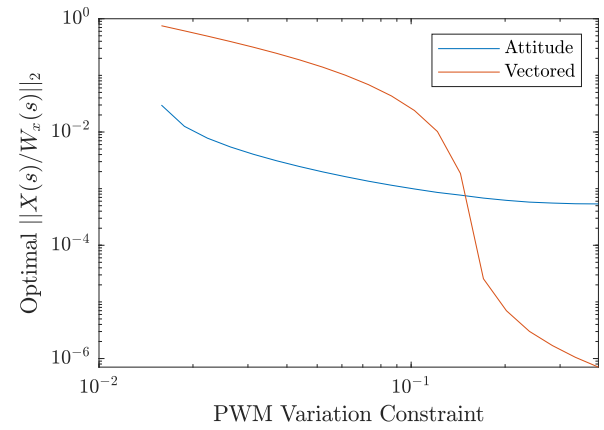


Figure 8: Variation in Optimal Norm, with actuation constraint, for the two actuation modes.

In general, the hard limit reached by attitude-based thrust can best be explained by looking at the pole-zero maps for $X(s)/F_{x,x}(s)$, shown in Figure 9.

It is observed in Figure 9 that the attitude-based thrust plant has zeros close to the imaginary axis. Ultimately this results in poor response of the system to actuation at this frequency, and the limit in attitude-based thrust is likely due to impossibly high levels of actuation which are required to effectively actuate this frequency notch.

This notch arises due to the fact that torque commands in the multirotor mixer (taken from [9]) are imperfect, and produce vectored thrust as a side effect. Whilst this could be fixed by changing coefficients, this mixer *leakage*, will inevitably occur with any linear multirotor mixer, as the operating point changes.

The vectored thrust leakage causes these zero issues, as when attitude-based thrust is used, a torque can only create translation through an angle change. As a result the acceleration due to attitude changes can roughly be described in Equation 6, where $\tau(s)$ represents the pitching torque.

$$s^2 X(s) \cong \frac{\alpha}{s^2} \tau(s) \quad (6)$$

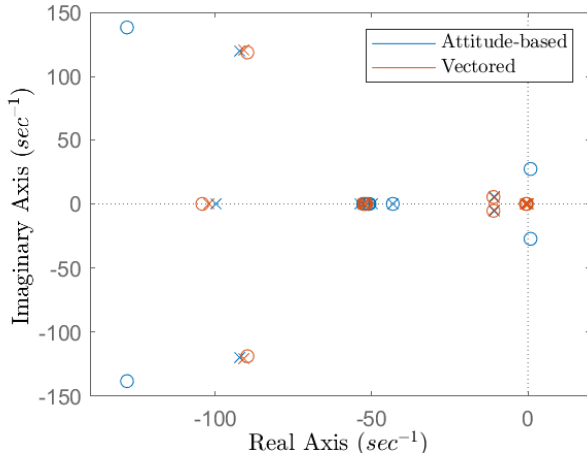


Figure 9: Pole Zero maps for $X(s)/F_x(s)$, with the two modes of actuation.

However, the leaked force will generate a force, as described in Equation 7.

$$s^2 X(s) \cong \frac{1}{m} \alpha_l \tau(s) \quad (7)$$

In Equation 7, α_l is a factor, describing how the leaked force is related to the applied torque. As a result, if α and α_l are of the same sign, the acceleration generated by the leaked force will be out of phase with the intended acceleration from the original torque command. When these are of the same magnitude, notches appear in the frequency response, which in turn creates problematic plant properties.

However, in tuning the controllers in simulation, this difference hasn't manifested in a significant manner. This is likely due to limitations in sensors and actuators, such as saturation and delay, which dominate over plant effects. For example, the simulation model of the extended Kalman filter is based on EKF, rather than the newer EKF2, available in the PX4 firmware [9]. Fixes to this may reveal aggressive tuning to be an area worth exploring.

6 CONCLUSION

This work has focused on identifying differences in how simple P-PID controllers can be tuned for wind disturbance rejection when various actuation methods are used on a fully-actuated octorotor. It has been found that by virtue of its less efficient force generation, vectored thrust tends to yield higher actuator usage when providing the same wind disturbance rejection performance, despite its high bandwidth. However, preliminary studies have shown that when the goal is a more aggressive tuning, attitude-based thrust control may be limited by its dynamics. Possible future work includes investigating comparison with more involved control methods. This work can also be extended to an experimental study.

ACKNOWLEDGEMENTS

The research reported was conducted as part of “Enabling unmanned aerial vehicles (drones) to use tools in complex dynamic environments UOCX2104”, funded by the New Zealand Ministry of Business, Innovation and Employment.

REFERENCES

- [1] Victoria Gonzalez-Dugo, P. Zarco-Tejada, E. Nicolás, Pedro Antonio Nortes, J. J. Alarcón, Diego S. Intrigliolo, and E.J.P.A. Fereres. Using high resolution UAV thermal imagery to assess the variability in the water status of five fruit tree species within a commercial orchard. *Precision Agriculture*, 14:660–678, 2013.
- [2] Ruimin Ke, Zhibin Li, Sung Kim, John Ash, Zhiyong Cui, and Yinhai Wang. Real-time bidirectional traffic flow parameter estimation from aerial videos. *IEEE Transactions on Intelligent Transportation Systems*, 18(4):890–901, 2017.
- [3] Chuang Deng, Shengwei Wang, Zhi Huang, Zhongfu Tan, and Junyong Liu. Unmanned aerial vehicles for power line inspection: A cooperative way in platforms and communications. *J. Commun.*, 9(9):687–692, 2014.
- [4] Hazim Shakhathreh, Ahmad H. Sawalmeh, Ala Al-Fuqaha, Zuochao Dou, Eyad Almaita, Issa Khalil, Noor Shamsiah Othman, Abdallah Khreishah, and Mohsen Guizani. Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges. *IEEE Access*, 7:48572–48634, 2019.
- [5] James R. Kutia, Karl A. Stol, and Weiliang Xu. Aerial manipulator interactions with trees for canopy sampling. *IEEE/ASME transactions on mechatronics*, 23(4):1740–1749, 2018.
- [6] Peter Zheng, Xinkai Tan, Basaran Bahadir Kocer, Erdeng Yang, and Mirko Kovac. TiltDrone: A fully-actuated tilting quadrotor platform. *IEEE Robotics and Automation Letters*, 5(4):6845–6852, 2020.
- [7] Jorge M. Arizaga, Herman Castañeda, and Pedro Castillo. Adaptive control for a tilted-motors hexacopter UAS flying on a perturbed environment. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 171–177, 2019.
- [8] Markus Ryll, Davide Bicego, and Antonio Franchi. Modeling and control of FAST-hex: a fully-actuated by synchronized-tilting hexarotor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1689–1694, 2016.
- [9] Jérémie Xavier Joseph Bannwarth. *Aerodynamic modelling and wind disturbance rejection of multirotor unmanned aerial vehicles*. PhD thesis, The University of Auckland, 2021.

- [10] Shi Lu, Armando Rodriguez, Konstantinos Tsakalis, and Yan Chen. Design and characteristics of a new transformable UAV with both coplanar and omnidirectional features. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 831–838, 2022.
- [11] Guangying Jiang and Richard Voyles. A nonparallel hexrotor UAV with faster response to disturbances for precision position keeping. In *IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 1–5, 2014.
- [12] Dario Brescianini and Raffaello D’Andrea. An omnidirectional multirotor vehicle. *Mechatronics*, 55:76–93, 2018.
- [13] Mahathi Bhargavapuri, Jay Patrikar, Soumya Ranjan Sahoo, and Mangal Kothari. A low-cost tilt-augmented quadrotor helicopter : modeling and control. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 186–194, 2018.
- [14] Michelangelo Nigro, Francesco Pierri, and Fabrizio Caccavale. Preliminary design, modeling and control of a fully actuated quadrotor UAV. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1108–1116. IEEE, 2019.
- [15] Rumi Kumar, Shubham R. Agarwal, and Manish Kumar. Modeling and control of a tethered tilt-rotor quadcopter with atmospheric wind model. *IFAC-PapersOnLine*, 54(20):463–468, 2021.
- [16] Murray L Ireland, Aldo Vargas, and David Anderson. A comparison of closed-loop performance of multirotor configurations using non-linear dynamic inversion control. *Aerospace*, 2(2):325–352, 2015.
- [17] G Bressan, D Invernizzi, S Panza, M Lovera, et al. Attitude control of multirotor uavs: cascade P/PID vs PI-like architecture. In *5th CEAS Specialist Conference on Guidance, Navigation and Control-EuroGNC*, pages 1–20, 2019.
- [18] Shilin Yi. *Modeling and control of an over-actuated quadrotor manipulator with four tiltable rotors*. PhD thesis, Okayama University, 2022.
- [19] Jorge A. Ricardo Jr and Davi A. Santos. Smooth second-order sliding mode control for fully actuated multirotor aerial vehicles. *ISA Transactions*, 129:169–178, 2022.
- [20] Maximilian Brunner, Karen Bodie, Mina Kamel, Michael Pantic, Weixuan Zhang, Juan Nieto, and Roland Siegwart. Trajectory tracking nonlinear model predictive control for an overactuated MAV. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5342–5348, 2020.
- [21] Panfeng Shu, Feng Li, Junjie Zhao, and Masahiro Oya. Robust adaptive control for a novel fully-actuated octocopter uav with wind disturbance. *Journal of Intelligent & Robotic Systems*, 103(6):1–17, 2021.
- [22] Tomoki Anzai, Moju Zhao, Masaki Murooka, Fan Shi, Kei Okada, and Masayuki Inaba. Design, modeling and control of fully actuated 2D transformable aerial robot with 1 DoF thrust vectorable link module. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2820–2826. IEEE, 2019.
- [23] Guangying Jiang, Richard M Voyles, and Jae Jung Choi. Precision fully-actuated UAV for visual and physical inspection of structures for nuclear decommissioning and search and rescue. In *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–7, 2018.
- [24] Z. J. Chen, K. A. Stol, and P. J. Richards. Preliminary design of multirotor UAVs with tilted-rotors for improved disturbance rejection capability. *Aerospace Science and Technology*, 92:635–643, 2019.
- [25] C. Probine, D. Yang, K. A. Stol, and N. Kay. Model predictive control on a fully-actuated octocopter for wind disturbance rejection. In *International Micro Aerial Vehicle Conference (IMAV)*, 2023.
- [26] Pierre Apkarian, Pascal Gahinet, and Craig Buhr. Multi-model, multi-objective tuning of fixed-structure controllers. In *European Control Conference (ECC)*, pages 856–861, 2014.

Distributed Nonlinear Model Predictive Control for a Quadrotor UAV

Bilal Mubdir* and Emmanuel Prempain
University of Leicester, University Rd, Leicester, UK

ABSTRACT

A Distributed Nonlinear Model Predictive Control (DNMPC) approach is proposed to control the simplified decoupled dynamics of a quadrotor UAV. The performance of DNMPC is compared, in terms of tracking and execution time, to that of standard control configurations based on centralized MPC and PID control aiming to show the suitability of each configuration in terms of performance and the practicality of using a particular configuration in real-time applications. The results show the advantage of using DNMPC in terms of ease of tuning and computational cost over more centralized feedback control approaches.

1 INTRODUCTION

In the last decades, different control techniques have been used in the aerospace industry. These range from basic classical control to modern artificial intelligence techniques used in many other fields. Classical controllers such as Proportional–Integral–Derivative PID [1], one of the best-known controllers, are successful in stabilizing a wide range of systems [2]. Aerospace systems such as aircraft or spacecraft, are nonlinear systems that could benefit from being controlled by advanced control systems enforcing strict stability and reliability requirements.

MPC is an advanced proactive control approach in which the control law is internally determined according to the forecasted response of the dynamic model to be controlled [2]. An open loop optimal control problem (OLOCP) is solved within a specified duration called Prediction Horizon to produce an optimal manipulated input trajectory for the optimal anticipated state trajectory. Out of the determined manipulated input trajectory, only the first control input action is applied to the system, and the whole process is repeated at every time sample. Furthermore, one of the major features of MPC is that it can handle multiple outputs and multiple inputs, or what is called MIMO systems [3], which makes it perfect for multivariable feedback control since the 1990s [4].

The testbed for this research is a Quadrotor, which is an Unmanned Aerial Vehicle (UAV) that features exceptional manoeuvrability, hovering, vertical take-off and landing capabilities.

MPC is extensively used in the literature as a flight controller for quadrotors. Identifying a suitable dynamic model for a quadrotor is crucial when it comes to MPC. The literature gives formulations of MPC for different model types, such as linear time-invariant, linear time-varying, piecewise affine, nonlinear, . . . etc. Since the mathematical model is the centrepiece of MPC, the model type is pivotal in how efficiently the OLOCP will be solved by the optimization solver. Complex dynamics representations may lead to a computational burden when solving the OLOCP.

Linear dynamics were considered in designing a linear model predictive control (LMPC) for the quadrotor in [5], resulting in a good tracking performance and disturbances rejection for different trajectories. Alexies et al. [6] used a switching type MPC in controlling the quadrotor but with a linearized piecewise affine model around multiple operating points to track a trajectory with the presence of disturbances. The authors decoupled the dynamics to form what they call it “dual control scheme.” A separate MPC was used to control the translational motion using the translational augmented dynamics, and another MPC was used to control the attitude of the quadrotor. As a matter of fact, the attitude dynamics (angular motion) of the quadrotor is faster in nature compared to the translational dynamics [6]. Thus, separating the controller for the translational motion and attitude is possible when the dynamics are decoupled.

A centralized nonlinear MPC was investigated in [7] based on a reduced nonlinear dynamic model of a quadrotor. It was shown that the use of a reduced nonlinear model led to a simpler MPC solution, easier to implement producing similar performance to that of an MPC solution based on a full order model. Reducing the computational complexity has been addressed effectively in [8] using machine learning to learn the control law from well-designed MPC flight data. However, this approach replaces the MPC with a clone that is limited by the conditions of the dataset used in the training stage.

Also, many researchers used the nominal MPC in conjunction with another technique. The authors in [9] proposed a two-loop controller in which the outer loop is MPC based to control the translational motion and a faster inner loop based on PID to control the attitude of the quadrotor. Although they have proved in simulation tests the effectiveness of using MPC to respect the constraints of the actuators and tracking the trajectory accurately, the disturbance effect was not introduced in their proposal. Hence, robustness concerns could

*Email address(es): bama4@le.ac.uk

arise, and the constraints may be violated if the controller is not robust enough when the quadrotor is subjected to a disturbance. Moreover, comparative studies were conducted to quantify the performance and the control effort of the linear MPC versus PID, PD, and Linear Quadratic Regulator (LQR) in the [10]. The authors claim that MPC offers better tracking performance compared to the other techniques. Their study adopted the centralized LMPC and tested on a smooth trajectory without any sharp manoeuvres. The results are convincing but when it comes to fast and very sharp manoeuvres, the results need to be verified.

The configuration of MPC formulation and how it handles the system dynamics could be varied according to the system complexity and the performance requirements [11]. In general, there are three different MPC configurations or architectures to control the system, namely; Centralized, Decentralized, and Distributed MPC. In centralized MPC, large-scale systems are controlled using one single MPC that has one optimal problem. Although this architecture is the greatest for performance, it is considered impracticable for large systems due to its computational requirements [12]. The decentralized MPC architecture features a separate MPC for each subsystem of the dynamics. Usually, the subsystem dynamics are decoupled from each other, the mutual interaction is ignored, as well as, each MPC has its optimal problem [11, 12]. On the other hand, the distributed model predictive control (DMPC) structure shown in Figure 1, is introduced when the decentralized configuration is associated with a communication ability between the individual MPC agents to facilitate exchanging information between them [12].

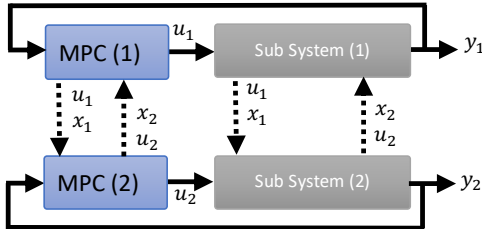


Figure 1: Typical configuration of distributed MPC.

The theory of DMPC and its stability has been extensively studied in the literature [13, 14]. Although the purpose of DMPC is to control large-scale with multiple objectives system, it has been applied successfully in controlling AC/AC converters in power applications to reduce the computational burden when centralized MPC is used [15]. The previous statement brings us to the contributions of the current research. To the best of our knowledge, no study was conducted to analyze the performance of different MPC configurations mainly when the nonlinear model is used. Furthermore, adopting DNMP for quadrotors has gained less interest in the UAV control literature. Therefore, the contri-

butions of this research are twofold:

- Introducing Distributed Nonlinear MPC for controlling the quadrotor by decomposing the centralized MPC into multiple agents whilst reducing the computation complexity of the overall controller.
- Evaluate various MPC configurations in terms of performance and computational complexity.

The remainder of this paper is organized as follows: Section 2 focuses on the derivation of the quadrotor UAV model and its dynamics as a part of the control system. Section 3 highlights the problem formulation and theory behind the Distributed Nonlinear MPC. Section 4 gives the design of the DNMP for the quadrotor, Section 5 demonstrates the different tests applied on the proposed DNMP and the other configurations and the results with a brief discussion while Section 6 concludes the findings of the paper.

1.1 Notation

In this paper, vectors are represented in bold lowercase letters (e.g., x), and the uppercase bolded letters are used for matrices (e.g., A). Other symbols represent scalars regardless of the letter case. The Euclidean Q -weighted norm is denoted by $\|x\|_Q^2 = x^T Q x$ where Q is a positive definite real symmetric matrix.

2 QUADROTOR DYNAMIC MODEL

To model the dynamics of the quadrotor and describe its states, it is crucial to define the coordinate systems. The inertial frame is fixed on the earth, entitled "Earth Frame". This frame denoted $\mathcal{F}_E$ with an x-axis directed to the north, a y-axis directed to the east, and a z-axis directed down toward the earth, while $\mathcal{F}_B$ represents the quadrotor as a rigid body frame as shown in Figure 2. The attitude of the quadrotor is defined by the Euler angles, roll angle ϕ , pitch angle θ , and yaw angle ψ . Due to the time independency of the quadrotor's body inertia and its symmetric structure, the motion equations can be achieved with respect to its body frame $\mathcal{F}_B$ [16]. The

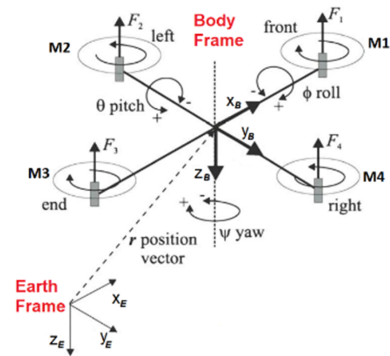


Figure 2: Frames coordinate systems.

absolute position of the quadrotor is $\xi = [x, y, z]^T$ and the attitude denoted by $\eta = [\phi, \theta, \psi]^T$. The model in this paper

follows the derivations made in [17] and [18]. The thrust T_i is aligned with the rotor axis i which rotates at angular velocity ω_i and is given by:

$$T_i = b\omega_i^2, \quad i \in 1, 2, 3, 4 \quad (1)$$

in which b is the lift constant equal to $C_T \rho A r^2$, where ρ is the density of the air that surrounds the propeller of surface area A and radius of r and where C_T is the thrust factor. Hence, the thrust along the z-direction of the quadrotor is given by:

$$T = b \sum_{i=0}^4 \omega_i^2 = b (\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \quad (2)$$

On the other hand, M_ϕ and M_θ are the pitching and rolling moments due to the thrust differences between the rotors. The net drag force due to the rotors is causing the heading moment M_ψ . The moments are given by:

$$M_\phi = l (T_4 - T_2) \quad (3a)$$

$$M_\theta = l (T_3 - T_1) \quad (3b)$$

$$M_\psi = d (-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \quad (3c)$$

where l is the distance from the quadrotor's COM to the rotor axis, d is the drag constant given by $C_P \rho A r^3$, and C_P is the torque coefficient of the motor. If quadrotor translational and angular velocity vectors are $\dot{\xi} = [U, V, W]^T$ and $\dot{\eta} = [P, S, R]^T$ respectively, the translational and angular motion equations are given by:

$$\begin{bmatrix} \dot{U} \\ \dot{V} \\ \dot{W} \\ \dot{P} \\ \dot{Q} \\ \dot{R} \end{bmatrix} = \begin{bmatrix} (\sin \phi \sin \psi + \cos \phi \sin \theta \cos \psi) \frac{T}{m} - \frac{A_x}{m} U \\ (-\sin \phi \cos \psi + \cos \phi \sin \theta \sin \psi) \frac{T}{m} - \frac{A_y}{m} V \\ -g + (\cos \phi \cos \theta) \frac{T}{m} - \frac{A_z}{m} W \\ \frac{1}{I_{xx}} \{(I_{yy} - I_{zz}) Q R - J_r \omega_T Q + M_\phi - A_r P\} \\ \frac{1}{I_{yy}} \{(I_{zz} - I_{xx}) P R - J_r \omega_T P + M_\theta - A_r Q\} \\ \frac{1}{I_{zz}} \{(I_{xx} - I_{yy}) P Q + M_\psi - A_r R\} \end{bmatrix} \quad (4)$$

where $\omega_T = -\omega_1 + \omega_2 - \omega_3 + \omega_4$, J_r is the motor's rotor inertia, A_r is the rotational aerodynamic drag coefficient and A_x , A_y and A_z are the linear aerodynamic drag coefficients in the x, y, and z directions respectively. Also, I_{xx} , I_{yy} , and I_{zz} denote the time-invariant inertia of the quadrotor about the body axes in $\mathcal{F}_B$. As discussed before, the only variables that can be controlled are the angular velocity of the motors. Before designing a control system, it is necessary to map the control signals to state equations. Thrust and moments are related to the motors angular velocities thanks to an allocation matrix, as follows:

$$\begin{bmatrix} T \\ M_\phi \\ M_\theta \\ M_\psi \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ 0 & -lb & 0 & lb \\ -lb & 0 & lb & 0 \\ -d & d & -d & d \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \quad (5)$$

The parameters of the quadrotor UAV used in this paper are

adopted from [19] and given in Table 1.

Symbol	Description	Value (Units)
g	Acceleration due to gravity	9.806 (m/s ²)
A_r	Rotational aerodynamic drag coefficient	10e-15 (Nm.s/rad)
m	Total mass of quadcopter UAV	0.65 (kgs)
l	distance from the quadrotor's COM motor	0.232 (m)
A_x, A_y, A_z	Linear aerodynamic drag coefficient	10e-15 (N.s/m)
J_r	Rotor inertia	4e-4 (kg.m ²)
I_{xx}	Moment of Inertia along x-axis	7.5e-3 (Nm.s ² /rad)
I_{yy}	Moment of Inertia along y-axis	7.5e-3 (Nm.s ² /rad)
I_{zz}	Moment of Inertia along z-axis	1.3e-2 (Nm.s ² /rad)
ρ	Air Density	1.293 (kg/m ³)
r	Propellers Radius	0.15 (m)
C_T	Thrust Coefficient	0.055
C_P	Torque Coefficient	0.024

Table 1: Quadrotor UAV parameters.

3 PROBLEM FORMULATION

Distributed Nonlinear MPC (DNMPC) is composed of N_a agents, where, each agent can be viewed as a local model predictive controller with a separate open-loop optimal control problem (OLOCP). Considering the following nonlinear discrete-time local system model for agent i [14],

$$\mathbf{x}_{k+1}^i = f_i(\mathbf{x}_k^i, \mathbf{u}_k^i), \quad i \in \{1, \dots, N_a\} \quad (6)$$

Where, $\mathbf{x}_k^i \in \mathbb{R}^{n_i}$ and $\mathbf{u}_k^i \in \mathbb{R}^{m_i}$ are the local state and control vectors of agent i respectively, n , and m representing the local state, and input vector dimensions, respectively. $\mathbf{x}_k^i$ is the successor or the next state and $f_i : \mathbb{R}^{n_i} \times \mathbb{R}^{m_i} \rightarrow \mathbb{R}^{n_i}$ is naturally defined by the local system differential equation that equals zero at zero initial conditions $f_i(0, 0)$. The sets of constraints that represent the safety limits and physical requirements for the local state and control inputs are denoted $\mathbb{X}^i \in \mathbb{R}^{n_i}$ and $\mathbb{U}^i \in \mathbb{R}^{m_i}$ respectively. They also represent the feasible state and input of the agent i . Therefore

$$\mathbf{x}_k^i \in \mathbb{X}^i, \quad \mathbf{u}_k^i \in \mathbb{U}^i, \quad k \geq 0, \quad i \in \{1, \dots, N_a\} \quad (7)$$

At time step k , the current measure state $\mathbf{x}_k^i$ is used to solve an OLOCP based on the local system model (6). Whatever the formulation of OLOCP, its solution is an optimal state trajectory that stabilizes the system in addition to an optimal control input trajectory that satisfies the anticipated state trajectory. Specifying the prediction horizon is vital to formulate the OLOCP problem and hence selecting the appropriate optimization solver. According to [12], the OLOCP problem is either infinite or finite. In this paper, the upper approximate finite OLOCP version adopted with the Terminal Cost function V_{f_i} , and Terminal Constraint $\mathbb{X}_{f_i} \in \mathbb{R}^{n_i}$ such that:

$$\mathbf{x}_N^i \in \mathbb{X}_{f_i}, \quad i \in \{1, \dots, N_a\} \quad (8)$$

The control objective is to use the receding horizon control approach to cooperatively stabilize the N_a agents and

converging the state of each agent to the equilibrium point $(\mathbf{x}_e^i, \mathbf{u}_e^i)$ by minimizing the following OLOCP

$$\min_{\mathbf{u}} J_k(\mathbf{x}_k, \mathbf{u}_k) = \sum_{k=0}^{N-1} \ell(\mathbf{x}_k, \mathbf{u}_k) + V_f(\mathbf{x}_N) \quad (9)$$

Where N is the prediction horizon, and

$$\mathbf{x}_k = [(x^1)^T, \dots, (x^{N_a})^T]^T, \quad \mathbf{x}_k \in \mathbb{R}^n, \quad n = \sum_i n_i \quad (10a)$$

$$\mathbf{u}_k = [(u^1)^T, \dots, (u^{N_a})^T]^T, \quad \mathbf{u}_k \in \mathbb{R}^m, \quad m = \sum_i m_i \quad (10b)$$

Then, the overall dynamics can be expressed as

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \quad (11)$$

where $f = [f_1, \dots, f_{N_a}]^T$, $f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ and the global equilibrium point for all agents is (x_e, u_e) . The OLOCP in (9) can be used in centralized MPC and is subjected to $\mathbb{X} = \mathbb{X}^1 \times \dots \times \mathbb{X}^{N_a}$, $\mathbb{U} = \mathbb{U}^1 \times \dots \times \mathbb{U}^{N_a}$, and $\mathbb{X}_f = \mathbb{X}_{f_1} \times \dots \times \mathbb{X}_{f_{N_a}}$. Although this formulation slightly increases the computation cost when it comes to solving the optimization problem, the terminal cost and terminal constraints assure the stability and feasibility of the controlled system [12, 20]. There is no specific rule to determine the terminal cost, however, it captures the cost beyond N up to ∞ to guarantee the full covering of the horizon [21].

The objective of each agent's optimization problem is to minimize the tracking error between the local state $\mathbf{x}_k^i$ and its reference trajectory $\bar{\mathbf{x}}^i$. Furthermore, to decompose the centralized MPC into a distributed MPC, the following two assumptions are made:

Assumption 1: The models for all agents presented in (6) are decoupled and the communication between agents is sufficient and available as needed by any agent.

Assumption 2: The condition $f_i(0, 0) = 0$ is not restrictive, since $f_i(x_e^i, u_e^i)$ is not zero, thus, the equilibrium point could be any arbitrary reference $\in \mathbb{X}_{f_i}$.

Accordingly, in a distributed style, problem (9) is divided into multiple OLOCPs given by

$$\begin{aligned} \min_{\mathbf{u}^i} J_k^i(\mathbf{u}_k^i, \mathbf{x}_k^i, \bar{\mathbf{x}}_k^i) &= \sum_{k=0}^{N-1} \ell_i(\mathbf{x}_k^i, \mathbf{u}_k^i, \bar{\mathbf{x}}_k^i) + V_{f_i}(\mathbf{x}_N^i) \\ \text{s.t. } (6), (7), (8), \quad \mathbf{x}_0^i &= \mathbf{x}_k^i \end{aligned} \quad (12)$$

Where the stage cost function $\ell_i(\mathbf{x}_k^i, \mathbf{u}_k^i)$ and terminal cost function $V_{f_i}(\mathbf{x}_N^i)$ are defined as:

$$\ell_i(\mathbf{x}_k^i, \mathbf{u}_k^i, \bar{\mathbf{x}}_k^i) = \|\mathbf{x}_k^i - \bar{\mathbf{x}}_k^i\|_{\mathbf{Q}_i}^2 + \|\mathbf{u}_k^i\|_{\mathbf{R}_i}^2 \quad (13a)$$

$$V_{f_i}(\mathbf{x}_N^i) = \|\mathbf{x}_N^i - \bar{\mathbf{x}}_N^i\|_{\mathbf{P}_i}^2 \quad (13b)$$

where $\mathbf{Q}_i$, $\mathbf{R}_i$, and $\mathbf{P}_i$ are positive definite weighting matrices for penalising the states, manipulated inputs, and terminal state, respectively for agent i . For each agent i , the control

law at the end of each prediction iteration is

$$\mathbf{u}_k^i = \mathbf{u}_0^{i*}(\mathbf{x}_k^i) \quad (14)$$

The superscript $*$ indicates the optimal value and $\mathbf{u}_0^i$ is the first control input action in the optimal control input trajectory obtained from the OLOCP solution that is applied to the system.

4 DNMPCC FOR QUADROTOR UAV

When designing a controller for quadrotors, different effects could be considered in the dynamics model, the reader is referred to [22] for a list of these effects. Including all effects with a full dynamics representation for the plant will intuitively lead to an increase in computational complexity. Therefore, a simplified version of the dynamics is used such as [7] to achieve faster solution during each prediction iteration. However, contrary to [9], the linear and rotational aerodynamic drag forces are considered in this paper. The resultant dynamics are driven from (4), decoupled and reduced based on the following assumptions:

Assumption 3: The gyroscopic effect of the motor is ignored.

Assumption 4: When the quadrotor is moving in the x-axis direction (pitching), the rolling angle is assumed zero.

Assumption 5: When the quadrotor is moving in a y-axis direction or rolling, the pitching angle is assumed zero.

Assumption 6: When the quadrotor moves vertically, the pitching and rolling angles are assumed zero.

Assumption 7: The heading of the quadrotor is assumed not controlled and the related torque is zero.

If T_s is the sampling time, the subscript k denotes the value of the variables at time kT_s , and by using the forward Euler discretization, the simplified decoupled dynamics are given as follows

$$\mathbf{x}_{k+1} = \begin{bmatrix} x_k \\ U_k \\ y_k \\ V_k \\ z_k \\ W_k \\ \phi_k \\ P_k \\ \theta_k \\ Q_k \end{bmatrix} + T_s \begin{bmatrix} U_k \\ m^{-1}(\sin \theta_k T_k - A_x U_k) \\ V_k \\ m^{-1}(-\sin \phi_k T_k - A_y V_k) \\ W_k \\ -g + m^{-1}(T_k - A_z W_k) \\ P_k \\ I_{xx}^{-1} M_{\phi_k} - A_r P_k \\ Q_k \\ I_{yy}^{-1} M_{\theta_k} - A_r Q_k \end{bmatrix} \quad (15)$$

Although the nonlinearity has been significantly reduced and weakened, the above dynamics in [14] without considering the linear and rotational aerodynamic drag forces, are validated in [7] when the pitching and rolling angles are limited to 0.2618 rad. Considering the linear and rotational aerodynamic drag forces, it is expected to perform better and acts more accurately compared to what was adopted in [7] with a little additional computation cost.

4.1 DNMPC Controller

Inspired by [15], with assumptions (1-7), we considered the Quadrotor UAV dynamics as a large-scale system in this paper, thanks to dynamics decoupling, (15) represented as five interconnected subsystems where each subsystem is controlled by its own MPC agent shown in Figure 3 to form the DNMPC with $N_a = 5$. Z-MPC, X-MPC, and Y-MPC form the first MPC agents' group for controlling the translational motion, z-axis, x-axis, and y-axis respectively. On the other hand, Rolling MPC, and Pitching MPC form the second MPC agents' group for controlling the angular position of ϕ and θ respectively.

The reference set-point is provided for each corresponding MPC agent in the first group. The role of the Z-MPC is to determine the optimal thrust force T , then feed it to the allocation matrix, X-MPC and Y-MPC as a cooperative input within the first group. In the proposed design, X-MPC and Y-MPC were utilized to generate the attitude trajectory for the second MPC agents' group. Thus, the two manipulated inputs ϕ , and θ , were the optimal inputs resultant from the Y-MPC, and X-MPC respectively and are provided to the second group as reference trajectory. Finally, the second group of agents, namely, Rolling MPC, and Pitching MPC, are designed to determine the optimal rotational torques, M_{ϕ_k} , and M_{θ_k} respectively and feeding them to the allocation matrix. The local subsystem's states and manipulated inputs for each MPC agent are summarized in the equations set (16). It is noteworthy to mention that according to assumption 7, no controller was used for controlling the heading of the Quadrotor UAV.

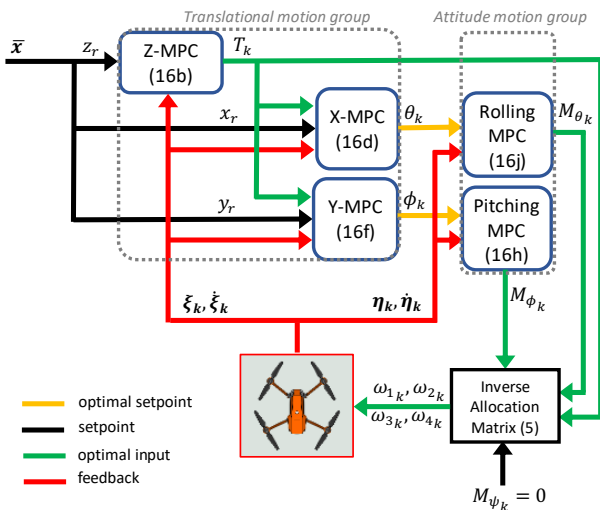


Figure 3: Block diagram for Distributed nonlinear MPC for Quadrotor UAV.

$$\mathbf{x}_k^1 = \begin{bmatrix} z_k \\ W_k \end{bmatrix}, \quad \mathbf{u}_k^1 = T_k \quad (16a)$$

$$\mathbf{x}_{k+1}^1 = \mathbf{x}_k^1 + T_s \begin{bmatrix} -g + m^{-1}(T_k - A_z W_k) \end{bmatrix} \quad (16b)$$

$$\mathbf{x}_k^2 = \begin{bmatrix} x_k \\ U_k \end{bmatrix}, \quad \mathbf{u}_k^2 = \theta_k \quad (16c)$$

$$\mathbf{x}_{k+1}^1 = \mathbf{x}_k^1 + T_s \begin{bmatrix} m^{-1}(\sin \theta_k T_k - A_x U_k) \end{bmatrix} \quad (16d)$$

$$\mathbf{x}_k^3 = \begin{bmatrix} y_k \\ V_k \end{bmatrix}, \quad \mathbf{u}_k^3 = \phi_k \quad (16e)$$

$$\mathbf{x}_{k+1}^3 = \mathbf{x}_k^3 + T_s \begin{bmatrix} m^{-1}(-\sin \phi_k T_k - A_y V_k) \end{bmatrix} \quad (16f)$$

$$\mathbf{x}_k^4 = \begin{bmatrix} \phi_k \\ P_k \end{bmatrix}, \quad \mathbf{u}_k^4 = M_{\phi_k} \quad (16g)$$

$$\mathbf{x}_{k+1}^4 = \mathbf{x}_k^4 + T_s \begin{bmatrix} I_{xx}^{-1} M_{\phi_k} - A_r P_k \end{bmatrix} \quad (16h)$$

$$\mathbf{x}_k^5 = \begin{bmatrix} \theta_k \\ Q_k \end{bmatrix}, \quad \mathbf{u}_k^5 = M_{\theta_k} \quad (16i)$$

$$\mathbf{x}_{k+1}^5 = \mathbf{x}_k^5 + T_s \begin{bmatrix} I_{yy}^{-1} M_{\theta_k} - A_r Q_k \end{bmatrix} \quad (16j)$$

4.2 Handling Constraint

In order to unify the framework of the analysis and introduce a fair comparison between the various configurations, unified constraints have been applied. The constraints set for the forces (5) are calculated based on the technical parameters of the brushless motors and their propellers. Assuming that all propulsion brushless motors are identical, the minimum lifting force is equal to mg , whereas the maximum lift force is given (2). Also, by recalling (3), therefore, the maximum and the minimum value of the rotational moment can be calculated by substituting the parameters of Table 2 in (3).

5 SIMULATION RESULTS & DISCUSSION

In this section, the simulating environment under which the proposed DNMPC controller and the other three different configurations are tested is explained. The other configurations are the Centralized Nonlinear MPC with full dynamics (CNMPC-F) built with dynamics (4), Centralized Nonlinear MPC with reduced decoupled dynamics (CNMPC-R) built with dynamics (15), and Nonlinear MPC with PID (MPC-PID) in [9].

5.1 Simulation Setup

The modelling of the quadrotor UAV has been done in MATLAB/Simulink while the model predictive controllers in all configurations have been carried out in MATLAB in MATLAB/Simulink by integrating CasADi, an open-source optimization toolkit with MATLAB interface [23]. The nonlinear OLOCP has been solved using the Interior Point Optimizer (Ipopt) [24], which is natively available in CasADi. The weighting matrices for each configuration, prediction hori-

zon, control horizon, and sampling time are summarized in Table 2.

CNMPC Sampling Time T_s	0.05 s
DNMPC Translation Group T_s	0.1 s
DNMPC Attitude Group T_s	0.05 s
MPC in MPC-PID T_s	0.05 s
PID Sampling time T_s	0.01 s
Prediction horizon	40
Weighting matrices for CNMPC	$\mathbf{Q} = \mathbf{P} = \text{diag}(15, 15, 150)$ $\mathbf{R} = \text{diag}(0.5, 8, 8, 0.01)$
Z-MPC weight in DNMPC	$\mathbf{Q} = \mathbf{P} = 10, \mathbf{R} = 0.09$
X-MPC weight in DNMPC	$\mathbf{Q} = \mathbf{P} = 35, \mathbf{R} = 50$
Y-MPC weight in DNMPC	$\mathbf{Q} = \mathbf{P} = 35, \mathbf{R} = 50$
Rolling MPC weight in DNMPC	$\mathbf{Q} = \mathbf{P} = 200, \mathbf{R} = 20$
Pitching MPC weight in DNMPC	$\mathbf{Q} = \mathbf{P} = 200, \mathbf{R} = 20$
Weighting matrices for CNMPC	$\mathbf{Q} = \mathbf{P} = \text{diag}(5, 5, 80)$ $\mathbf{R} = \text{diag}(0.1, 7, 7)$
Z-MPC weight in DNMPC	$\mathbf{Q} = \mathbf{P} = 10, \mathbf{R} = 0.09$
PID controller parameters	$KP_\theta = KP_\phi = 25$ $KI_\theta = KI_\phi = 0.1$ $KD_\theta = KD_\phi = 15$

Table 2: Controllers parameters for all configurations.

5.2 Step Response Results

All configurations were tested by applying a step input of 5m on the x-axis while the quadrotor is at a hovering point (0, 0, 4m). The step response characteristics for each configuration have been summarized in Table 3.

	CNMPC-F	CNMPC-R	DNMPC	MPC-PID
Rise Time	1.4207	1.567	2.1466	2.3924
Transient Time	2.5409	2.7869	3.4209	3.509
Settling Time	2.5409	2.7869	3.4209	3.509
Settling Min	4.5022	4.5288	4.5387	4.5608
Settling Max	5.0001	5.0001	5.0000	5.0159
Overshoot	0.0017	0.0015	0	0.3157
Peak	5.0001	5.0001	5.0000	5.0159
Peak Time	5.65	5.40	7.00	4.15

Table 3: Step Response characteristics for each configuration.

5.3 Tracking Results

Two trajectories were used to test the tracking performance of each configuration. Figures (4) and (5) illustrate the tracking response for the infinity shape and the multiple short rectangles' trajectory respectively. The corresponding motors' angular velocities for each tracking response are illustrated in Figures (6) and (7). It is seen how the DNMPC configuration provides fewer changes for all trajectories, especially for the sharp manoeuvres. Moreover, it is obvious that the MPC-PID configuration has the worst variation, and this is expected as the only optimal input is the left force, where the other forces are generated from the PID. To investigate the practicability and performance of each configuration, the solution time for the OLOCP in each prediction cycle. In addition, the root mean square error RMSE has been determined for all trajectories. Table 5.3 shows the solution time and RMSE of each configuration for the tested trajectories.

Infinity Shape Trajectory (40 s)				
RMSE x-axis (m)	0.3556	0.29982	0.29688	0.22453
RMSE y-axis (m)	0.35302	0.30021	0.2982	0.23771
RMSE z-axis (m)	0.40636	0.40672	0.53857	0.29981
Total Solving time (s)	68.3392	14.9918	25.0127	10.8356
Average Solving time (ms)	85.42	18.74	31.22	13.54

Multiple Short Rectangles' Trajectory (26 s)				
RMSE x-axis (m)	0.96495	1.0071	1.0551	1.0633
RMSE y-axis (m)	0.68085	0.67747	0.71234	0.712
RMSE z-axis (m)	0.42955	0.42866	0.52551	0.38389
Total Solving time (s)	36.0099	14.7204	17.6784	9.0684
Average Solving time (ms)	69.25	28.30	33.93	17.43

Table 4: Step Response characteristics for each configuration.

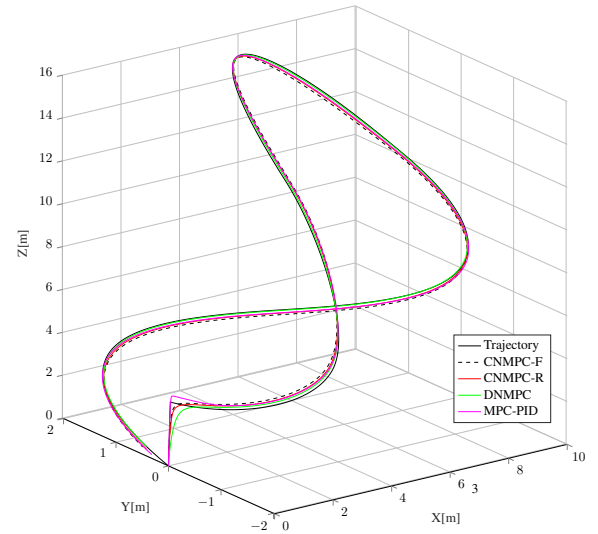


Figure 4: Tracking response for an infinity shape trajectory.

5.4 Discussion

It is seen from Table 5.3 that decomposing the dynamics and reducing its complexity in the DNMPC led to a significant computation time reduction. That is expected as computational complexity and thus the computation cost can be reduced when the problem dimension is scaled down, for instance, the factorization in Newton-type solver requires $2n^3$ flops (floating point operations) for a state equation of dimension n [25]. Therefore, when scaling down dynamics (4) used in the CNMPC-F to dimension 2 in the individual state equations of (16), the number of flops required for factorization is then reduced to 1/216 (where n divided by 6). Concerning tuning, the different configurations have been tuned by a simple strategy where all weighting parameters are initiated with 1, and then manipulated in steps until an acceptable step response performance is achieved. As a matter of fact, tuning the centralized MPC is challenging. However, the DNMPC configuration was the most flexible one for tuning. This is because each agent in the proposed DNMPC is independent of the other in terms of its cost function and penalty weight.

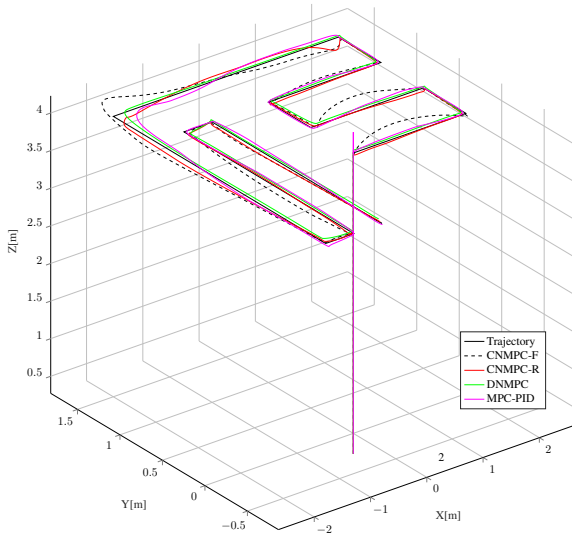


Figure 5: Tracking response for multiple short rectangles' trajectory.

The results reveal that the MPC-PID configuration is faster than the other configurations since the time needed for solving the OLOCP in its MPC is 84% better than the nominal CNMPC-F configuration. In terms of practicability and real-time application, the CNMPC-R, DNMPC, and MPC-PID configurations are more suitable than using the CNMPC-F. The step response characteristics in Table 3 provide a plain understanding of the response for each configuration and it is discernible that the DNMPC is slower compared to the other configurations but steadier in reaching the set-point.

6 CONCLUSION

This paper explores the possible configuration for designing MPC based flight controller for a quadrotor UAV. The dynamics of the quadrotor UAV were modelled using its equations of motion and used as an oriented model for the MPC. Four configurations were studied based on two architectures, centralized and distributed architecture. A nonlinear flight controller based on the distributed MPC was proposed, in which five separate MPC agents were used to stabilise the quadrotor. All configurations were tested with a step input in addition to tracking two different trajectories. The results disclose that the use of simplified decoupled dynamics reduces the computational complexity of the overall control law. The proposed DNMPC unfolds promising performance, especially for the sharp manoeuvres trajectories. Furthermore, the DMPC is more flexible and easier to tune than the conventional MPC. Further investigations could be conducted to evaluate the practicability and the suitability of each configuration, where real-time optimization is limited by the capabilities of the on-board microcontroller.

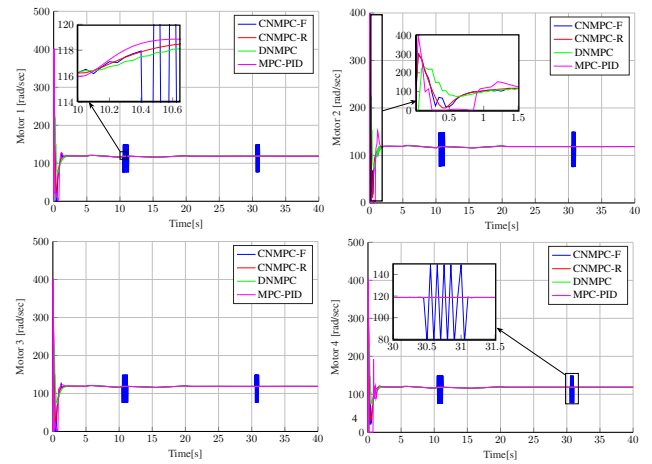


Figure 6: Motors' angular velocities for an infinity shape trajectory.

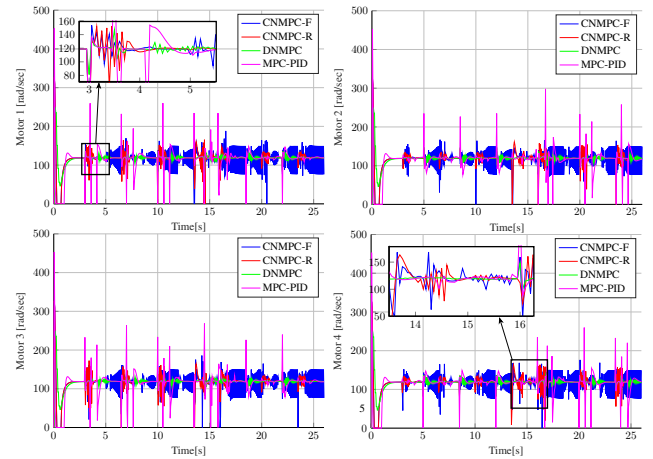


Figure 7: Motors' angular velocities for multiple short rectangles' trajectory.

ACKNOWLEDGEMENTS

This research is funded by the Higher Committee for Education Development in Iraq (HCED) and supported by Sulaimani Polytechnic University, Sulaymaniyah, Iraq.

REFERENCES

- [1] Ang Kiam Heong, G. Chong, and Li Yun. Pid control system analysis, design, and technology. *IEEE Transactions on Control Systems Technology*, 13(4):559–576, 2005.
- [2] Max Schwenzer, Muzaffer Ay, Thomas Bergs, and Dirk Abel. Review on model predictive control: an engineering perspective. *The International Journal of Advanced Manufacturing Technology*, 117(5-6):1327–1349, 2021.

- [3] C. H. F. Silva, H. M. Henrique, and L. C. Oliveira-Lopes. Experimental application of predictive controllers. *Journal of Control Science and Engineering*, 2012:1–18, 2012.
- [4] Manfred Morari. Model predictive control: Multivariable control technique of choice in the 1990s? *Oxford science publications*, 1993.
- [5] M. Islam, M. Okasha, and M. M. Idres. Dynamics and control of quadcopter using linear model predictive control approach. *IOP Conference Series: Materials Science and Engineering*, 270, 2017.
- [6] K. Alexis, G. Nikolakopoulos, and A. Tzes. On trajectory tracking model predictive control of an unmanned quadrotor helicopter subject to aerodynamic disturbances. *Asian Journal of Control*, 16(1):209–224, 2014.
- [7] M. Abdolhosseini, Y. M. Zhang, and C. A. Rabbath. An efficient model predictive control scheme for an unmanned quadrotor helicopter. *Journal of Intelligent & Robotic Systems*, 70(1-4):27–38, 2012.
- [8] Guanrui Li, Alex Tunchez, and Giuseppe Loianno. Learning model predictive control for quadrotors. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 5872–5878, 2022.
- [9] Huan Cheng and Yanhua Yang. Model predictive control and pid for path following of an unmanned quadrotor helicopter. In *2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 768–773, 2017.
- [10] Mohamed Okasha, Jordan Kravev, and Maidul Islam. Design and experimental comparison of pid, lqr and mpc stabilizing controllers for parrot mambo mini-drone. *Aerospace*, 9(6), 2022.
- [11] Yang Shi and Kunwu Zhang. Advanced model predictive control framework for autonomous intelligent mechatronic systems: A tutorial overview and perspectives. *Annual Reviews in Control*, 52:170–196, 2021.
- [12] Utku Eren, Anna Prach, Başaran Bahadır Koçer, Saša V. Raković, Erdal Kayacan, and Behçet Açıkmeşe. Model predictive control in aerospace systems: Current state and opportunities. *Journal of Guidance, Control, and Dynamics*, 40(7):1541–1566, 2017.
- [13] William B. Dunbar. Distributed receding horizon control of dynamically coupled nonlinear systems. *IEEE Transactions on Automatic Control*, 52(7):1249–1263, 2007.
- [14] Wei Shanbi, Chai Yi, and Li Penghua. Distributed model predictive control of the multi-agent systems with improving control performance. *Journal of Control Science and Engineering*, 2012:1–8, 2012.
- [15] Luca Tarisciotti, Giovanni Lo Calzo, Alberto Gaeta, Pericle Zanchetta, Felipe Valencia, and Doris Saez. A distributed model predictive control strategy for back-to-back converters. *IEEE Transactions on Industrial Electronics*, 63(9):5867–5878, 2016.
- [16] Zoran Benic, Petar Piljek, and Denis Kotarski. Mathematical modelling of unmanned aerial vehicles with four rotors. *Interdisciplinary Description of Complex Systems*, 14(1):88–100, 2016.
- [17] Hossein Bolandi, Mohammad Rezaei, Reza Mohsenipour, Hossein Nemati, and S. M. Smailzadeh. Attitude control of a quadrotor with optimized pid controller. *Intelligent Control and Automation*, 04(03):335–342, 2013.
- [18] Harikrishnan Suresh, Abid Sulficar, and Vijay Desai. Hovering control of a quadcopter using linear and nonlinear techniques. *International Journal of Mechatronics and Automation*, 6(2/3), 2018.
- [19] Armando S. Sanca, Pablo J. Alsina, and Jesús de Jesús F. Cerqueira. Dynamic modelling of a quadrotor aerial vehicle with nonlinear inputs. In *2008 IEEE Latin American Robotic Symposium*, pages 143–148, 2008.
- [20] J.B. Rawlings, D.Q. Mayne, and M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, second edition, 2020.
- [21] Tor Aksel N. Heirung, Joel A. Paulson, Jared O’Leary, and Ali Mesbah. Stochastic model predictive control — how does it work? *Computers & Chemical Engineering*, 114:158–170, 2018.
- [22] S. Norouzi Ghazbi, Y. Aghli, M. Alimohammadi, and A. A. Akbari. Quadrotors unmanned aerial vehicles: A review. *International Journal on Smart Sensing and Intelligent Systems*, 9(1):309–333, 2016.
- [23] Joel A. E. Andersson, Joris Gillis, Greg Horn, James B. Rawlings, and Moritz Diehl. Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.
- [24] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2005.
- [25] Rien Quirynen. *Numerical Simulation Methods for Embedded Optimization*. Dissertation, University of Freiburg, 2017.

Adaptive frequency control strategy for the propulsion management of a long-endurance drone using hybrid power sources

T. Pavot^{1*}, R. Kiefer^{1†}, T. Mesbahi¹, E. Laroche²

¹INSA Strasbourg, laboratoire ICube UMR7357, 24 Boulevard de la Victoire, Strasbourg, France

² Université de Strasbourg, laboratoire ICube UMR7357, 3-5 rue de l'université, Strasbourg, France

ABSTRACT

Multi-source electric hybridization is increasingly being considered for long-endurance UAVs, offering a credible alternative to traditional internal combustion engine-powered models. By using a combination of different energy sources, such as fuel cells, supercapacitors, batteries and an energy management algorithm, hybridization optimizes the lifetime of the drone's energy sources while reducing its environmental impact..

This paper proposes an innovative frequency strategy for power management in hybrid-powered UAVs by adapting the cut-off frequency according to the state of charge of the supercapacitors, in order to protect the fuel cell and battery from flight-induced current variations. This article shows the advantages of the adaptive frequency strategy over the conventional frequency strategy found in the literature.

1 INTRODUCTION

It currently seems essential to work on decarbonized propulsion strategies to limit greenhouse gas emissions. In the field of aeronautics, the aim is to create more efficient and less polluting propulsion systems. Long-endurance UAVs provide a testing and demonstration ground for developing innovative solutions.

Hybrid systems using fuel cells and batteries offer electrical alternatives to thermal technologies in terms of endurance. Over the past ten years, the development of hydrogen-powered solutions has gained momentum as a credible alternative to fossil fuels in terms of flight autonomy. In 2009, the US Navy introduced one of the first hybrid-powered UAVs, the Ion Tiger [1], equipped with a fuel cell and battery, capable of flying for 26 hours on gaseous hydrogen and 48 hours on liquid hydrogen. More recently, in 2018, Andrew Gong achieved the first flight with passive triple hybridization, using a fuel cell (FC), supercapacitor (SC) and battery. [2].

These advances testify to the growing interest in hybridization in this sector. However, the lifetime of fuel cells and batteries remains a major challenge for embedded applications.

The aim of active hybridization with supercapacitors (SC) and fuel cell (FC) is to optimize the use of each energy source to reduce wear and tear, in particular by controlling the dynamics of the currents demanded. This requires the development of high-performance energy management algorithms (EMS).[3].

The frequency strategy is a type of EMS that is highly developed in the literature. [4, 5]. The innovation proposed in this article makes the cut-off frequency variable as a function of the SC state-of-charge (SoC). To evaluate these strategies, performance indicators will be proposed.

The Stork drone (Fig1) used in this study was designed by the INSA Strasbourg team as part of the European INTERREG ELCOD project (2017-2020, www.elcod.eu). It has been specially designed to carry out long-duration flights for various missions, including the monitoring of atmospheric pollutants. The drone has a maximum take-off weight of 25 kg and a wingspan of 5 m.



Figure 1: Stork drone MK.I

The first part of the article will describe the chosen electric hybridization architecture. The second part will develop the choice of performance indicators in order to evaluate one strategy against another. The third section compares a conventional fixed-frequency energy management strategy with an adaptive-frequency strategy. Finally, we'll make a com-

*Email address(es): thomas.pavot@insa-strasbourg.fr

†Email address(es): renaud.kiefer@insa-strasbourg.fr

parative assessment of these two strategies.

2 HYBRIDIZATION ARCHITECTURE

There are different architectures for combining these energy sources. The chosen solution is the semi-active architecture [6], which uses only two converters (Fig2). It integrates three power sources for propulsion (Fuel Cell (FC), supercapacitor (SC) and batteries). These sources must be capable of supplying 500 W in cruise flight at a speed of 90 km/h, and 2000 W at peak, particularly during the take-off phase. The aim is to combine these sources with high-performance power electronics to create a hybrid propulsion system capable of responding to power peaks while maximizing the service life of the energy sources used.

By controlling these current converters, it is possible to regulate the energy supplied by each of the sources. The battery, on the other hand, is directly connected to the DC voltage bus to power the drone at start-up and provide additional power during take-off. This approach stabilizes the DC bus voltage around that of the battery, providing security in the event of DC-DC converter failure. The converters used here are limited to 500 W to meet the needs of cruising flight and for reasons of space inside the drone.

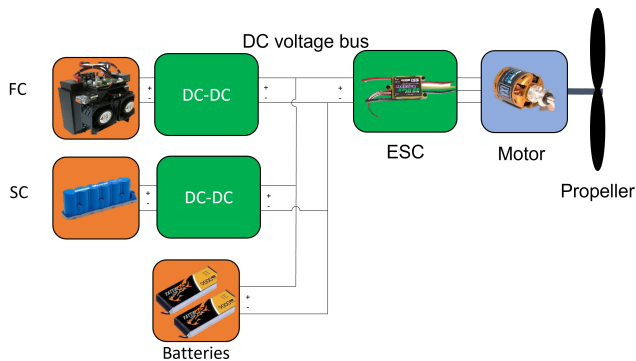


Figure 2: Semi-active architecture of the Stork drone

3 PERFORMANCE INDICATORS

To operate these converters efficiently, the EMS (Energy Management Strategy) is responsible for providing the current setpoint for the converters, to ensure that the energy mix required to meet the system's needs is achieved at all times.

In the case of UAVs, different types of EMS are available [7]. These strategies are designed to minimize or maximize one or more criteria. The choice of criteria is crucial to the evaluation of different EMS.

3.1 Typical performance indicators

For hybrid fuel cell systems, hydrogen consumption and service life are often the key criteria for comparing different EMS.

The literature often uses the hydrogen consumption criterion to evaluate EMS. However, hybrid systems combine batteries and supercapacitors with the fuel cell. Both are capable of storing energy. We therefore need to ensure that the hydrogen savings consumed do not come from discharging these sources. This criterion will not be used here, as the model used is too simplistic and does not allow us to distinguish a real difference in consumption according to current cycles.

The second criterion often used is ageing. As with hydrogen consumption, a precise model is needed to provide a reliable prediction of fuel cell lifetime.

Recent publications report on fuel cell ageing as a function of charge cycle. It appears that road cycles wear out faster than constant-current cycles. Tests have been carried out to see the impact between two cycles with different dynamics [8]. The conclusion is that the less dynamic cycle leads to slower ageing than the other. Similarly, M. Tognan's PhD thesis [9] which studies the ageing of four single-cells in a FC, two of which are directly paralleled with an SC. The effect of the SC is to smooth current calls. The results on ageing are striking. The two hybridized single cells wear less than the other two (degradation rate around 2 times lower). These results show that the reduction in current cycle dynamics on the FC has a direct impact on ageing.

However, it is difficult to evaluate these parameters accurately in simulation. The performance indicator we're going to use corresponds to the reduction in current variations demanded of the fuel cell.

3.2 Selected performance indicators

We have chosen these three indicators for our study :

- The first indicator compares the root mean square (RMS) of the current variations required from the FC over the entire current cycle. This is the indicator that quantifies the ageing of the FC [8, 9].

- The second indicator measures the SC's maximum operating range. To minimize current peaks in the FC and the battery, SC must be used to their full capacity. To respect the optimal operating range, it's essential not to go below 25% of the SC's SoC. [10]. This choice is arbitrary, but it ensures that the SC voltage doesn't drop too low (half the voltage) to limit joule losses in the converters. This means that the maximum usable SoC range is 75%. This indicator is an important limit for validating the effectiveness of the strategy.

- The third indicator concerns the amount of energy passing through the SC. This is the sum of the energy absorbed and the energy supplied by the SC. This indicator serves as a control to assess the impact of SC use over the entire flight cycle.

These three indicators will enable us to compare strategies.

3.3 Reference cycle

In order to be able to compare the strategies used, a reference current cycle was defined during a flight involving

several altitude levels (100m, 150m and 200m).(Fig.3). The drone had a total mass of 16 kg and flew for 24 min without FC. To simulate the mass of the FC and its tank, the drone was weighted by 4 kg and carried a payload of around 2 kg. This flight provided data on the drone's power consumption. It was found that the current could reach up to 80 A during ascent phases, while at cruising speed, the average consumption was 16 A, i.e. around 500W. (Fig.4). This figure shows the current peaks associated with changes in altitude.

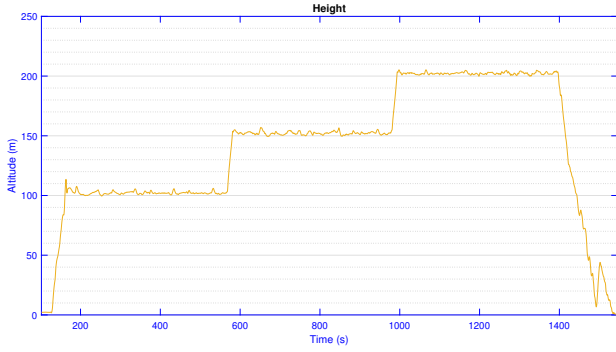


Figure 3: Altitude change during reference flight

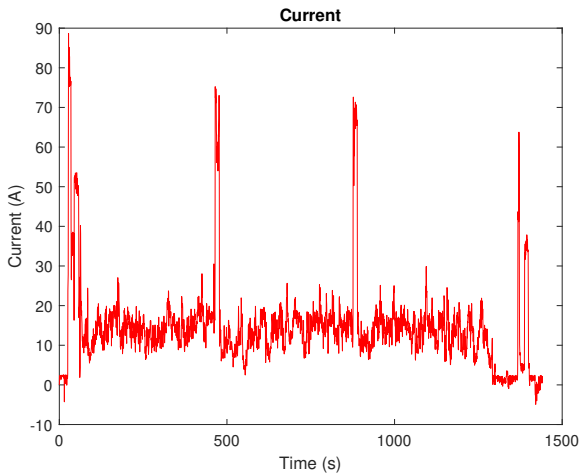


Figure 4: Power consumption during flight

This current reference cycle will be integrated into our simulation models in order to compare our different algorithms.

4 ENERGY MANAGEMENT STRATEGY

The strategies adopted here are based on the frequency strategy [4, 5]. This type of strategy reduces current variations from one source to another. On our system, this type of EMS reduces the FC wear factor.

4.1 Fixed frequency strategy

The frequency strategy consists of using a low-pass filter to divide the current drawn by the load into two setpoint currents for the FC and SC DC-DC converters. The FC current comprises the low frequencies, while the SC current comprises the high frequencies. This approach saves the battery and the FC during cruising flight by using the SC to supply the current variations. It is necessary to determine a suitable cut-off frequency that will limit FC current variations while respecting SC limits. However, the ideal setting will depend on the flight cycle. This setting must avoid excessively discharging or overloading the SC, while maintaining a sufficiently slow response time to minimize FC ageing.

In the following, we'll use a fixed "ideal" cutoff frequency as a reference value. This ideal frequency is obtained by applying the maximum current step that the converters can withstand. The ideal frequency limits the SC's state-of-charge (SoC) drop to 25% from an initial value of 50%. It is therefore the most dynamic fixed cut-off frequency during these steps that keeps the SC in its optimum operating range. In the case of our application, a current step of 16 A is applied. The ideal fixed cut-off frequency obtained is $FC = 0.2rad/s$.

This fixed frequency is kept as low as possible to limit the dynamics of the current required from the fuel Cell.

4.2 SC state-of-charge (SoC) adaptive frequency strategy

The innovation proposed here consists in adjusting the cutoff frequency according to the SoC of the SC. The general idea is to take advantage of the variation amplitude of the SC's SoC to maintain a high dynamic range, thus limiting current variations on the FC.

Depending on the operating mode of the SC (charging or discharging), the frequency law is different. When the SC is discharging, the system should be less sensitive if the SoC is low, and more sensitive if its SoC is high. The opposite is true during charging. To address this issue, we have chosen a cutoff frequency equation that follows an exponential law.

The variable-frequency strategy is based on the use of two symmetrical equations that link the SoC of the SC to the cut-off frequency. The two equations (Eq 1 and 2) used to relate the SoC to the cutoff frequency (f_c) are symmetrical exponential functions centered on an SoC of 0.5. These equations are defined on the interval [0;1]. The first equation is used for discharging the SC, while the second equation is used for charging it. The cut-off frequency chosen depends on the direction of variation of the load current in order to select the equation.

$$f_c(\text{SoC})_{\text{discharge}} = (F_{cHlim} - F_{cLlim}) \cdot e^{-G \cdot (\text{SoC})} + F_{cLlim} \quad (1)$$

$$f_c(\text{SoC})_{\text{charge}} = (F_{cHlim} - F_{cLlim}) \cdot e^{-G \cdot (1-\text{SoC})} + F_{cLlim} \quad (2)$$

The equation comprises three parameters: F_{cHlim} , F_{cLlim} and G (Tab. 1). The first two parameters are chosen to define the frequency range of the equation. The aim is

to stay within the dynamic range of the slowest source, in this case the FC, and to achieve a very high dynamic range for the SC. The G parameter is also crucial (Fig5), as it determines the steepness of the exponential equation. Once the limits are defined, this parameter adjusts the sensitivity of the equation.

Table 1: Tattu LiPo Batteries

Paramètres	Valeur
F_{cHlim}	1 rad/s
F_{cLlim}	0.01 rad/s
G	3, 5 et 10

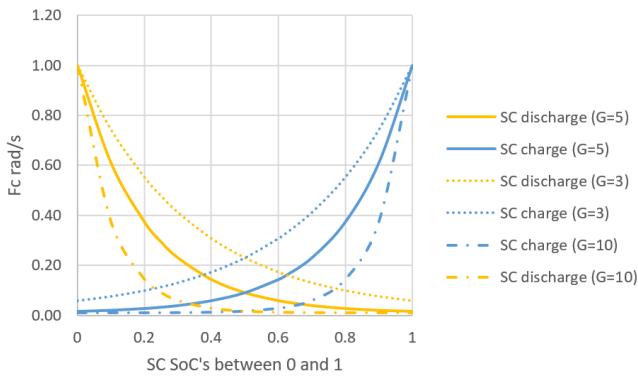


Figure 5: Representation of cut-off frequency as a function of SoC and parameter G

5 SIMULATION AND RESULTS

Modeling and simulations are carried out using Matlab Simulink software, applying the current cycle derived from real drone flight. The relevance of the simulation model of the various EMS components was validated by comparing the results with those of a test bench emulating the drone's flight in the laboratory, using the cycle defined above.

5.1 Simulation results

The results obtained (Fig 6) highlight the differences between the ideal fixed-frequency strategy of 0.2 rad/s and three strategies with adaptive cutoff frequencies, with different G parameter values (3, 5 and 10). By changing the G parameter, the sensitivity of the cut-off frequency to the SC SoC changes.

The first curve represents the current drawn by the load, which must be distributed between the three sources. The next two curves correspond to the currents coming from the FC and the SC at the output of the converters, while the last curve represents the battery current. Each of these strategies smoothes the current demanded from the FC, while leaving current variations to the SC. The battery draws current during climb phases, as the power demanded by the motor exceeds

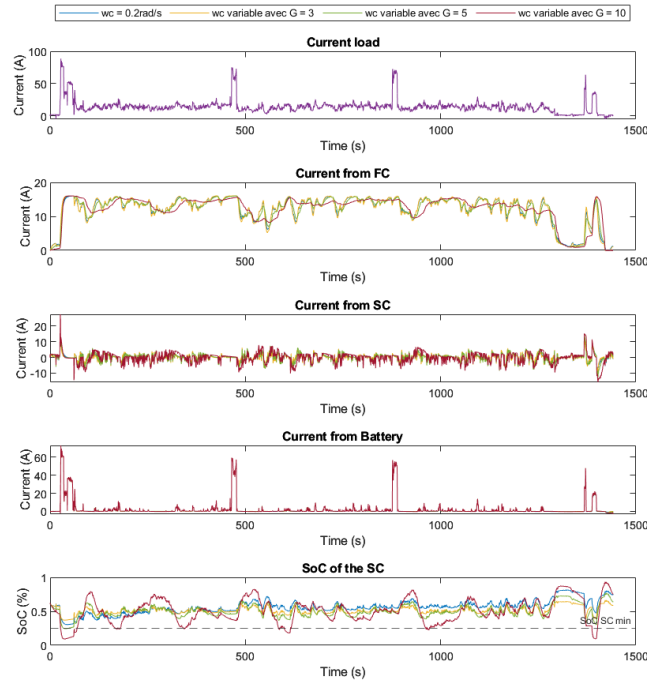


Figure 6: SC current and SoC results during the complete flight current cycle

500 W, whereas during cruise phases, the current is zero. This is due to the limitation of the converters' power to 500 W.

The last curve illustrates the evolution of the SC's SoC throughout the cycle. It is interesting to note the effect of parameter G , which increases the SC's charging and discharging amplitude as its value increases.

The choice of three values of G demonstrates the impact of sensitivity on simulation results. Value 5 is selected to illustrate the optimum result, which keeps the SC SoC above 25% throughout the cycle. The other two values allow us to observe the evolutionary trend of the G parameter around this optimum value.

The performance indicators (Fig 7) allow us to evaluate the different strategies. The strategy with $G=10$ in yellow is the worst performer on all three indicators. The strategy with $G=3$ in red appears to be the best, but results in SC discharge below 25%, exceeding the set limit. The other two curves, blue (fixed frequency) and green (variable frequency with $G=5$) correspond, according to our sizing protocol, to the ideal strategies for a fixed and adaptive cut-off frequency, as they approach the 25% limit without exceeding it.

It can be clearly seen that, over the whole flight, the adaptive frequency strategy in green ($G = 5$) improves two indicators and equals the third compared with the fixed-frequency strategy in blue. The most notable reduction was in current variation for the FC, down 31%. Energy transiting through the SC also increased by around 16%. The amplitude of the SoC remains virtually unchanged for both strategies.

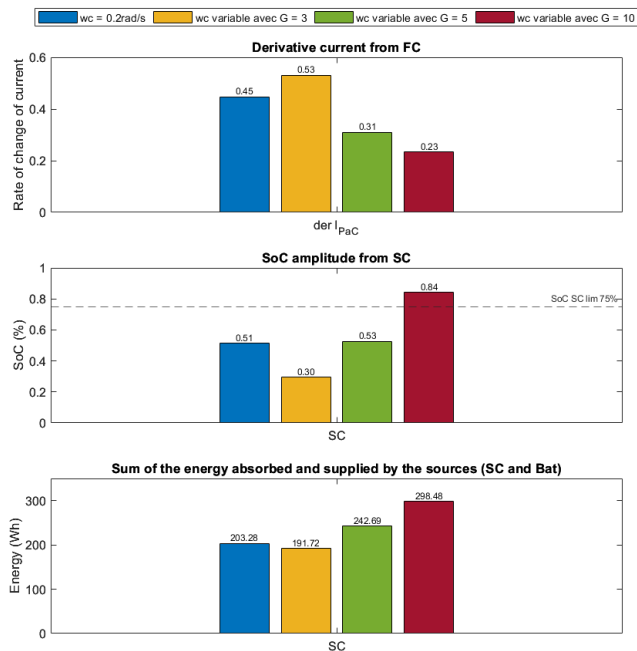


Figure 7: Performance indicators by strategy

6 DISCUSSION

The use of an adaptive frequency leads to an improvement in the indicators presented above compared with the fixed-frequency strategy. However, the best indicator should be the gain in lifetime of one strategy over another, which requires more detailed FC modeling, an aspect still little explored in the literature. An important assumption made here is that reducing current variations extends the lifetime of fuel cells. It is with this in mind that this new strategy has been developed. Utilization of the SC SoC amplitude, which remains around 50%, has not been maximized. Use of up to 75% could improve the other indicators. To apply this new strategy to any hybrid system with the same architecture, it is necessary to develop a sizing protocol to determine the three parameters of the equation. The current approach, which uses the maximum current step the system can withstand, is a first method, but it may not be optimal, especially for oversized systems. A final, more general point concerns the impact of hybridization on the lifetime versus autonomy of a single-source solution. In the case of drones, flight endurance is an essential factor. Adding this hybridization leads to an increase in mass with the addition of the two converters and the SC, which reduces autonomy. In addition, the frequency-adaptive strategy encourages maximum use of the SC, which means that as much energy as possible passes through the converters, resulting in additional losses and a reduction in overall efficiency.

In the context of today's applications, drones tend to have a relatively short lifespan compared with the energy sources that power them. This can be explained by the rapid pace

of technological progress and frequent replacement cycles in the drone industry, where new, more advanced models are regularly introduced to the market.

However, when considering longer-term prospects in the aeronautics field, it becomes interesting to seek to develop aircraft with a longer lifespan. This would optimize investments in aircraft design, manufacture and maintenance, maximizing their use over an extended period.

7 CONCLUSION

This paper highlights an adaptive frequency strategy that shows promising potential for optimizing the use of SC in a hybrid system designed for a high-endurance UAV. Simulation results show that this strategy significantly reduces current variations at the fuel cell and battery terminals, and should therefore improve battery life.

Further research and experimental testing will be required to confirm this approach on test benches and in real flights.

Ultimately, this study opens up new possibilities for improving the performance and energy efficiency of long-endurance UAVs. Optimizing the use of the SC using an adaptive frequency strategy offers interesting prospects for aeronautical applications, enabling better management of current variations and more efficient use of power sources.

ACKNOWLEDGEMENTS

European INTERREG ELCOD project (2017-2020, www.elcod.eu)

REFERENCES

- [1] K Swider-Lyons, J Mackrell, J Rodgers, G Page, M Schuette, and R Stroman. Hydrogen fuel cell propulsion for long endurance small UAVs. *AIAA Centennial of Naval Aviation Forum "100 Years of Achievement and Progress"*, (September), 2011.
- [2] 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.
- [3] T Lei, Z Yang, Z Lin, and X Zhang. State of art on energy management strategy for hybrid-powered unmanned aerial vehicle. *Chinese Journal of Aeronautics*, 32(6):1488–1503, 2019.
- [4] H Marzougui, A Kadri, M Amari, and F Bacha. Energy management of fuel cell vehicle with hybrid storage system: A frequency based distribution. pages 1853–1858, 2019.
- [5] A Tani, M Camara, and B Dakyo. Energy management based on frequency approach for hybrid electric vehicle applications: Fuel-cell/lithium-battery and ultracapacitors. *IEEE Transactions on Vehicular Technology*, 61(8):3375–3386, 2012.

- [6] 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*, 116:100620, 2020.
- [7] M Boukoberine, M Zia, M Benbouzid, Z Zhou, and T Donato. Hybrid fuel cell powered drones energy management strategy improvement and hydrogen saving using real flight test data. *Energy Conversion and Management*, 236:113987, 2021.
- [8] 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.
- [9] M Tognan. Etude de dégradations des performances de Piles à Combustible PEM BT alimentées en H₂/O₂ lors de campagnes d'endurance: du suivi de l'état de santé en opération à la modélisation du vieillissement. pages 1–190, 2018.
- [10] L Mamouri, T Mesbahi, P Bartholomeus, and T Paul. Design of a DC/DC power converter for li-ion battery/supercapacitor hybrid energy storage system in electric vehicles. *2020 IEEE Vehicle Power and Propulsion Conference, VPPC 2020 - Proceedings*, (1), 2020.

Model Predictive Control on a Fully-actuated Octocopter for Wind Disturbance Rejection

Caleb Probine, David Yang, Karl A. Stol*, and Nicholas Kay

University of Auckland, Department of Mechanical and Mechatronics Engineering, Auckland 1010, New Zealand

ABSTRACT

Physical interaction with the outdoors is a key application of UAVs. However, in these situations aircraft must compensate for external disturbances such as wind. Fully-actuated UAVs are well suited to this, but consideration is needed in control design. This work focuses on how Model Predictive Control (MPC) can be applied here. MPC has the advantage of being able to take system constraints into account, and its utility is investigated in wind rejection tasks. This work has focused on the formulation of appropriate constraints on craft-orientation and motor efforts. Also incorporated are frequency dependent actuator weightings, and basic disturbance estimation. Overall, in simulation, modelling of motor constraints was not found to be effective when compared to prior controllers. However, the use of MPC with angle constraints was found to improve position regulation. The controller was then verified with wind tunnel experiments.

1 INTRODUCTION

In recent years, Unmanned Aerial Vehicles (UAVs) have rapidly increased in their civil applications [1]. Examples of use cases include agriculture [2] or medical device supply [3]. This work in particular focuses on multirotor UAVs, which are advantageous for their agility and vertical takeoff ability. Furthermore, unlike fixed-wing UAVs, they can hold a position, which allows for physical interaction with the environment. This interaction has a range of implications, and it allows the use of UAVs in scenarios such as canopy sampling [4]. However, one major concern in these tasks is wind disturbances. Not only do winds have a mean component to resist, but also, unpredictable stochastic components.

One method by which rejection of these stochastic components can be eased is by using fully-actuated UAVs. Fully-actuated UAVs can produce forces and torques in any direction, and as such, translation and rotation of the aircraft are decoupled. This contrasts more conventional quadrotors, which must tilt to translate. Tilting is limited by the rotational inertia of the aircraft, which limits the bandwidth of this actuation. Meanwhile, fully-actuated UAVs and their ability to

directly generate lateral forces, have significantly higher actuation bandwidths, making disturbance rejection easier. One type of fully-actuated drone is the fixed-tilt UAV, on which rotors are canted at constant angles. An example is shown in Figure 1, and this work will focus on these fixed-tilt UAVs.

A range of authors have already developed wind disturbance rejection for fully-actuated UAVs, and one example is the work in [6]. This describes a robust adaptive controller for a fully actuated octocopter similar to that shown in Figure 1. To account for disturbances, force-torque disturbances are estimated and used for regulation. The controller is then validated in simulation with a 'gust function'. While the controller shows good regulation ability, there are some limitations in the methodology. Firstly it is unclear whether the 'gust function' is backed up by literature. Furthermore, there are no experiments. However, these limitations are not unique to this work. The authors of [7] implement active disturbance rejection control on a passively-tilted craft, and similarly, they focus on simulation, using simple functions to generate force histories representing 'wind shear' or 'wind blow'. Meanwhile, the authors of [8] explore the application of sliding mode control for a tilting-rotors quadcopter. In this case, CFD is used to estimate the worst case disturbances, which are in turn used in simulation. However, this work again lacks experiments, and furthermore, the CFD methodology is not presented. Finally, the authors of [9], have designed another sliding mode controller for a tilting-rotors quadcopter. In the control design, torque disturbance effects are ignored, and a cuboid is used for the aerodynamic models. However, the use of mean wind, and Dryden turbulence models to generate wind profiles, does lend more validity. Nonetheless, experimental results are still not presented.

In the aforementioned work, a trend can be seen in fully-actuated UAV literature to only use simulation for the testing of wind rejection, with varying vigor. However, experimen-



Figure 1: Fixed-tilt Octocopter, utilised in [5]

*Email address: k.stol@auckland.ac.nz

tal validation is obviously of high importance. One work that provides this is [5]. Its author has designed an $\mathcal{H}_\infty$ controller for wind rejection on a fully-actuated octocopter. Experimental testing is then carried out in a wind tunnel to validate the controller's station keeping performance and shows improvements when compared to more traditional cascaded PID controllers. This work is not without issues though. One example, is that in simulation, some scenarios required re-tuning of the controller to avoid saturation of motor Pulse Width Modulations (PWMs). In the literature, there do exist other methods of experimental wind disturbance testing. For example, the authors of [10] fly aircraft outside. However, methods such as this, are limited by their repeatability. Meanwhile, the authors of [11] develop Iterative Learning Control for a fully-actuated craft and use air blowers to generate wind disturbances. However, in this case, details aren't present regarding wind affected area, or turbulence qualities. Across the works that have developed wind-rejecting controllers for fully-actuated UAVs, it is worth noting that none of them have explicitly dealt with the problem of saturation or constraints.

To handle the issue of constraints, one possibility is Model Predictive Control (MPC). MPC is an established control method, which relies on online optimisation. At each time step, a discrete plant model is used to gauge the response of the system to inputs. These control inputs are then chosen to minimise a cost function, and optimise future performance.

One of the critical advantages of MPC, is its ability to handle constraints, by incorporating them into the optimisation problem. This has been used on fully-actuated crafts in various works. The work in [12] constrains the forces and torques that the UAV produces in order to model the craft-limits. However this is possibly problematic in that the limits of each motor under combinations of forces may not be properly represented. The work in [13] provides an alternative solution to the actuator saturation problem. The authors of [13] discuss the fact that MPC applications on multirotors often apply a cascade control structure, with an unconstrained attitude controller. In these works constraints are then placed on state variables such as velocity or acceleration. Rather, in [13] a single monolithic Model Predictive Controller is used to control motors individually. Constraints are then applied to propeller rates for each rotor. A similar technique is utilised in [14] with constraints on propeller velocities. However, state constraints can still be valuable when, for example, obstacles need to be avoided in a trajectory, as shown in [15].

MPC can also be adapted to compensate for disturbances, via a range of techniques, some examples of which are given here. A common method, is the addition of control components for disturbance estimation. In [16], a wind disturbance is modelled as being driven by normally distributed noise. The disturbance is then estimated by an Extended Kalman Filter (EKF), and provided to the Model Predictive Controller. A similar approach is also taken in [12], where an EKF is used to compensate for disturbances and model

uncertainty. Techniques other than Extended Kalman Filtering have also been used in works such as [17], where an Extended State Observer is used. One interesting point in [17], is that integral components are also incorporated into the Model Predictive Controller to reject slow moving disturbances. Finally, works such as [18], represent a more holistic incorporation of disturbances. In [18], robust MPC is used, in which, the optimisation is modified to minimise the cost, while considering the worst case disturbances. However, methods such as this can be overly conservative and complicated.

Overall, the work in [5] provides a strong basis for the continuation of wind rejection experiments for fully-actuated UAVs. However, constraints, such as motor saturation, remain an area to be explored. The literature has provided a range of methods to handle constraints with MPC. The main contribution of this paper is the implementation and experimental testing of a Model Predictive Controller on a fixed-tilt UAV, under wind disturbances. This facilitates investigation of the ability of MPC to improve constrained performance.

2 MODELLING

The UAV utilised is shown in Figure 1. It has a flying mass of 1.72 kg and a rotor-to-rotor diameter of 500 mm. It is technically over-actuated due to its 8 rotors, but the focus in this work is on the features coming from full-actuation.

2.1 Co-ordinate Systems

Shown in Figure 2, a reference frame attached to the UAV-body is denoted by $\mathcal{B}\bullet$. Positive $\mathcal{B}[x, y, z]^\top$ denote the front, right and down directions respectively. Meanwhile $\mathcal{W}\bullet$ defines the inertial frame in which the UAV's position can be perceived. For the wind, the simplifying assumption is also made that the dominant component is in the $\mathcal{W}_x$ direction.

2.2 Octocopter Dynamics

As discussed earlier, there are two ways of actuating this craft. *Attitude-based thrusts* arise due the horizontal force produced when the UAV is tilted. The desired components of the net vector, are denoted as $\mathcal{W}F_A$, and this is produced by manipulating roll ψ and pitch θ . The unique translational forces that an Octocopter can generate are then discussed as *Vectored thrust* ($\mathcal{B}F_{V,x,y}$). In this case, rotors are selectively sped up to generate a net horizontal force. Given the higher bandwidth, it is desired to use vectored thrust to reject high frequency wind components. Meanwhile, attitude-based thrust should ideally be used to reject other components of the wind, given its better efficiency. These actuation methods are shown in Figure 2. Because attitude-based thrust is to be used for low frequency wind components, and mean wind is assumed to exist in the $\mathcal{W}_x$ axis alone, roll control will be ignored in this work. Rather, any $\mathcal{W}_y$ regulation will be done with $\mathcal{B}F_{V,y}$. This work also assumes ψ (yaw) deviation is negligible, such that the $\mathcal{B}_{xz}$ and $\mathcal{W}_{xz}$ planes are coincident throughout control. Note that this does not assume $\tau_z = 0$, but rather that the yaw regulation is perfect.

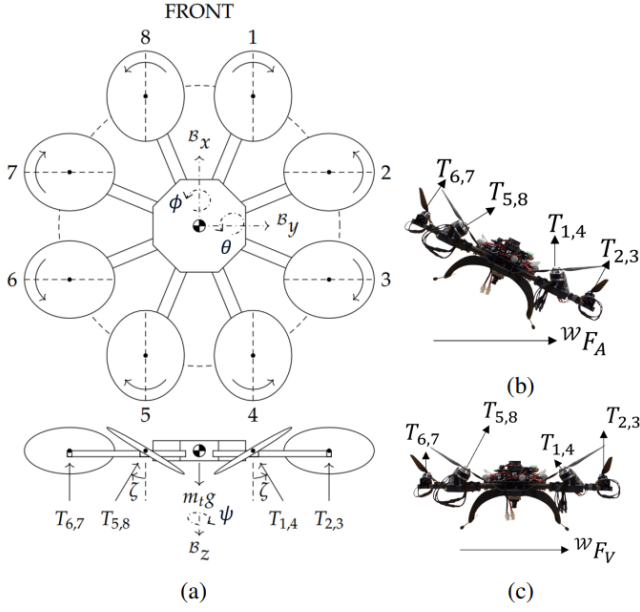


Figure 2: Craft dynamics illustrations. (a) Octocopter Schematic [5]; (b) Attitude-based thrust ${}^W F_A$ in the ${}^W y$ direction; (c) Vectored thrust ${}^B F_V$ in the ${}^B y$ direction

2.3 Existing Control Architecture and Model Components

The numerical model of the UAV is taken from [5], and includes a motor model, rigid body dynamics model and aerodynamic correlations. While detailed, limitations of this model include omitted dynamics, due to the empirical nature of the aerodynamic models, and unmodelled interactions between adjacent rotors due to cross flow. Alongside the physical components, the low-level control elements also need to be considered, as they are to be incorporated in the overall plant model that MPC uses. This includes the baseline attitude controller, as well as a motor mixer. Figure 3 then shows how all of these plant model components are combined.

This allows the designation of inputs for the model predictive controller to manipulate, as follows. The first two inputs are ${}^W F_{A,x}$ and ${}^W F_{A,z}$. Together these define the attitude-based thrust vector to be produced. They are converted to a single ${}^B z$ thrust (the magnitude) and an angle setpoint θ . The Attitude Controller utilised here is the default from the PX4 flight stack, and it generates the torque vector for the aircraft. Vectored thrusts ${}^B F_{V,x}$ and ${}^B F_{V,y}$ are also directly manipulated by the model predictive controller. Manipulation in the body frame is used to ease the conversion of these values to motor efforts. The body to world frame conversion is then encoded via the linearisation, and any effects should be small near the operating point. From the torques and vectored thrusts supplied as commands to the UAV, the Multirotor Mixer then generates PWMs for each of the motors. For this craft, the mixer is simply a matrix $\mathcal{M} \in \mathbb{R}^{8 \times 6}$

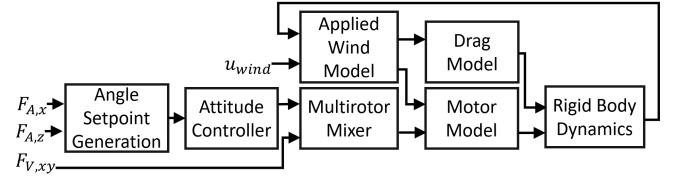


Figure 3: Model to be controlled by the Model Predictive Controller, developed in [5]

3 MODEL PREDICTIVE CONTROL FORMULATION

The aforementioned model is included in the controller as a discrete time state space model, for prediction. At each time step, a cost function is minimised, for the next N_p time-steps. To achieve this objective, the first N_c control actions are manipulated. When an optimal control sequence is determined, the first-time-step control action is applied to the plant. Constraints are then incorporated, by defining them as a linear inequality of inputs and outputs, with the format in Equation 1. In this equation $\mathbf{u}$ and $\mathbf{y}$ are input and outputs of the plant, $\mathbf{G}$ is a vector of constraint boundaries and E/F are matrices that encode the constraint coefficients.

$$E\mathbf{u} + F\mathbf{y} \leq \mathbf{G} \quad (1)$$

In MATLAB and Simulink, a suite of tools are used to setup the optimisation. As such this section details the formulation of appropriate architectures, constraints and models.

3.1 Controller Architecture

Mentioned earlier was the use of vectored and attitude-based thrusts at appropriate frequencies (Section 2.2). A potential method is the integration or filtering of a single control input to allocate between vectored and attitude-based thrust. However, the induced coupling between the thrusts would make constraint satisfaction difficult. Consequently, they were controlled independently. To add frequency dependent allocation, auxiliary states for the low-passed vectored thrust were added to the controller and weighted. This allowed for the penalisation of mean-value build-up in the vectored thrust. Meanwhile, the attitude-based thrust was controlled via an integral component. As such the output of the controller was $\frac{d}{dt} F_{A,x}$. By weighting the change in attitude-based thrust, high frequency components could effectively be penalised. This control architecture is shown in Figure 4.

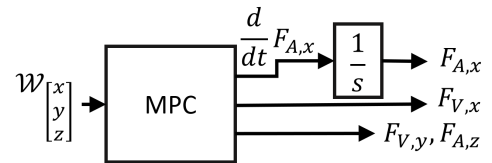


Figure 4: Model Predictive Controller Architecture

3.2 Constraints

3.2.1 Motor Constraints

The motivation for motor constraints arises in [5], in which, a range of different conditions necessitated re-tuning of the developed $\mathcal{H}_\infty$ to avoid violation. To formulate constraints, each motor was constrained individually, similar to [13]. However, in the case of this work, PWMs are constrained rather than propeller rates. This is because PWM saturation was highlighted in [5]. The core issue here was to determine how to appropriately constrain the PWMs, when they were not directly controlled. To do so, the motor map was used, allowing constraint formulation as shown in Equation 2. In Equation 2, the PWMs are on a normalised scale of $[0, 1]$.

$$0 \leq \mathcal{M} \begin{bmatrix} {}^B C_\eta \\ {}^B T \end{bmatrix} \leq 1 \quad (2)$$

In this equation ${}^B C_\eta$ and ${}^B T$ are the vectors of torques and forces respectively, produced by the craft. However, not all forces and torques are generated by the Model Predictive Controller. The first two elements of ${}^B T$, ($F_{V,x}$ and $F_{V,y}$) are manipulated directly by the controller. However, the other elements must be approximated or measured.

One assumption can be made with respect to roll and yaw. They have zero references, and so do not contribute to craft-translation. In fact, in simulation the roll and yaw angle values have standard deviations less than 0.2° . As such, the roll and yaw torques are assumed not to significantly affect the remaining dynamics, and they are decoupled in the model. They are then simply measured by the Model Predictive Controller, and input to the constraints.

${}^B T_z$ can also be easily approximated. This thrust is generated from $F_{A,z}$ and $F_{A,x}$, by a two-norm operation. This can simply be linearised about an operating point. Similarly, while the pitch torque is not generated by MPC, it is related to $F_{A,x}$, $F_{A,z}$ and the internal states of the attitude controller. As such, the pitch torque can be approximated from the estimates of the attitude controller states. The overall relationships are formulated in Equation 3.

$$0 \leq \begin{bmatrix} \mathcal{M}_{\tau_x}^\top \\ \mathcal{M}_{\tau_y}^\top \\ \mathcal{M}_{\tau_z}^\top \\ \mathcal{M}_{T_x}^\top \\ \mathcal{M}_{T_y}^\top \\ \mathcal{M}_{T_z}^\top \end{bmatrix}^\top \begin{bmatrix} \tau_x \\ c_1 F_{A,x} + c_2 F_{A,z} + \mathbf{C}_1 \hat{\mathbf{x}}_{\text{PID}} \\ \tau_z \\ F_{V,x} \\ F_{V,y} \\ c_3 F_{A,x} + c_4 F_{A,z} \end{bmatrix} \leq 1 \quad (3)$$

In Equation 3, c_1 , c_2 , $\mathbf{C}_1$, c_3 and c_4 are linearisation constants. Also note that $F_{V,x}$ and $F_{V,y}$ are outputs of the controller, while τ_x and τ_z are measured. Finally, $\hat{\mathbf{x}}_{\text{PID}}$ represents the internal state estimates for the attitude controller.

3.2.2 Angular Constraints

While not an initial concern in [5], another constraint considered was the aircraft orientation. An orientation limit may be motivated by a tool being used, or a requirement to keep a low vertical profile. As the wind is assumed to be coming from the $\mathcal{W}_x$ direction, the focus was on pitch constraints (θ).

This was considerably easier to incorporate into the control architecture, compared to PWM. The pitch angle was already a state in the model, and was constrained between two limits. The pitch angle setpoint was also constrained. Similar to the ${}^B T_z$ thrust, the pitch setpoint was directly related to $F_{A,z}$ and $F_{A,x}$. This relationship was again linearised to generate the pitch set-points of the aircraft, and subsequently constrained. It is worth noting that these constraints were *soft*. This is because the controller was not necessarily able to directly manipulate θ , as it was a state variable, rather than a control input. The use of soft constraints defines a limit, with a significant cost being imposed if said limit is exceeded.

3.3 Linearisation and Model Generation

One requirement for the Model Predictive Controller generation was a linear model to predict from. From [5], a non-linear model was already available. This had also been linearised in [5], but to account for new plant outputs, this was repeated. The linearisation point was found by trim analysis. This was carried out with an input wind speed of $\mathbf{u} = [5.6 \ 0 \ 0] \text{ m/s}$, and with the constraint that attitude-based thrust was exclusively being used to reject this mean wind. The wind velocity was chosen for consistency with results in [5]. Edits were then made to include features such as the low passing of vectored thrust and decoupling of yaw and roll torque measurements from controller states. A minimal realisation was also taken to eliminate one state that did not effect the weighted or constrained dynamic behaviour.

3.4 Disturbance Estimation

In the MPC toolbox used, a Kalman Filter was generated automatically. This allowed estimation of unmeasured components such as the attitude controller's internal states. However, this tended to fail under non-zero-mean disturbances, including both deviations of the mean wind from 5.6 m/s , and biases in sensor measurements. To compensate, simple disturbance estimators were added. These were formulated as auxiliary plant model states, which were in turn estimated by the Kalman Filter. These auxiliary states were modelled as integrators, driven by gaussian noise, shown in Equation 4.

$$\dot{x}_{aux} = B w_d \quad (4)$$

For input disturbances, the estimated state, is then fed into the state space model as shown in Equation 5.

$$\dot{x} = A x + B u + B_d x_{aux} \quad (5)$$

Meanwhile, for output disturbances the auxiliary state is simply added to the respective output as a bias.

The wind is modelled as an input disturbance, and the B_d matrix maps how a change in wind speed from the equilibrium point impacts the plant. This matrix was found during linearisation, and it allows the controller to perceive the effects of mean wind deviations. Meanwhile, output disturbances were included on the measured pitch angle, and the world frame velocities, to improve state estimation quality. Under preliminary simulation, disturbance estimation was found to help prevent $\mathcal{W}_x$ steady state errors. In this sense, the disturbance estimator acted similarly to an integral component in the controller. However, it has the dual advantage of also improving state estimate accuracy overall.

3.5 Final Parameters

The sample time was chosen by comparing the Model Predictive Controller's computational requirements to a known significant load on the PX4 flight stack in simulation. This informed a sample rate choice of 10 Hz. To determine the control and prediction horizons, they were chosen by balancing performance (as measured by root mean square (RMS) position error) and the allowable computational load. This resulted in horizons of $N_c = 3$ and $N_p = 8$.

In testing, the baseline controller for testing is the $\mathcal{H}_\infty$ controller developed in [5]. As such, the Model Predictive Controller weights were tuned by running it, and the $\mathcal{H}_\infty$ controller under wind, and ensuring that the standard deviations of each actuation method: e.g. $F_{V,x}$ and $F_{A,x}$, were similar. The power spectral densities were also verified to ensure they were fairly similar in shape. The weights manipulated were those on velocity, vectored thrust, and attitude-based thrust-rate. The velocity weight was increased mainly due to perceived resonance issues during preliminary experimental implementations. Meanwhile, throughout tuning, changing the low pass portion of the weighting did not significantly affect frequency allocation, as the normal vectored thrust weight was sufficient to generate the required effect. The final parameters chosen are summarised in Table 1. Note that scale factors were also used to improve numerical conditioning.

x, y, z	$F_{V,x}$	$F_{V,y}$	$F_{A,z}$	$\dot{x}, \dot{y}, \dot{z}$	$\dot{F}_{A,x}$	F_{xlp}
1	6.66	4	3.3	1.2	0.06	0.1

Table 1: Model Predictive Controller weights

4 TESTING METHODOLOGY

4.1 Simulation Methodology

The simulation is based on the non-linear model described in Section 2. Also included for simulation is a sensor model, as well as input wind trajectories. The trajectories utilised are also taken from [5], and are comprised of generated turbulent wind data with a mean speed of 5.6 m/s. The turbulence intensities are 10.03%, 8.28% and 8.02% for the $\mathcal{W}_x$, $\mathcal{W}_y$ and $\mathcal{W}_z$ axes respectively. At the beginning of each simulation a 10s period is allowed for the wind to ramp up.

To test PWM constraints in simulation, actuator saturation was induced for each controller. For $\mathcal{H}_\infty$, saturation is induced between the mixer and motors (Their relative positions in simulation are as shown in Figure 3). Meanwhile, for MPC, the optimisation's perception of the constraints is instead tightened. Under different saturation levels, the simulation is run for 120s, and the data is inspected to determine if position diverges. To impose angle constraints with $\mathcal{H}_\infty$ the pitch component of the angle setpoint was saturated. For MPC, the angle limits were updated in the optimisation. With angle limits, position variation was used for comparison.

4.2 Experimental Methodology

To convert the Model Predictive Controller for use with the PX4 flight stack, Simulink code generation was used. This generated a C++ class, with methods to execute each control step. This class was then added to the flight stack as its own module. Using radio-controlled switches, the MPC and Default controllers could then be switched as appropriate. The flight hardware used was a Pixhawk 4 controller.

The experimental results themselves came from the closed loop wind tunnel shown in Figure 5. For UAV flight, this was equipped with a motion capture system to provide accurate position feedback. A turbulence grid was also installed in the tunnel, and induced time varying components in the flow. The turbulence intensities in each axis are 11.6%, 6.2% and 7.3% for the $\mathcal{W}_x$, $\mathcal{W}_y$ and $\mathcal{W}_z$ axes respectively.



Figure 5: University of Auckland Closed Loop Wind Tunnel, with a turbulence grid and motion capture system.

5 RESULTS

5.1 Unconstrained Performance

5.1.1 Simulation

First tested was an unconstrained Model Predictive Controller. This was simulated for the purposes of verifying that actuator allocation was occurring as designed. Figure 6 shows the variation of attitude-based and vectored thrust based actions for the tuning described in section 3.5.

While it can be seen that attitude-based thrust is successfully used to compensate for mean wind, the vectored thrust seems to have less variation at all frequencies. This is due to the match to the actuator usage in [5]. However, a weight modification, shown in Figure 7, demonstrates that more appropriate actuator usage is possible. In this figure, vectored

thrust is shown to dominate at high frequencies. However, further experiments retain the matched tuning.

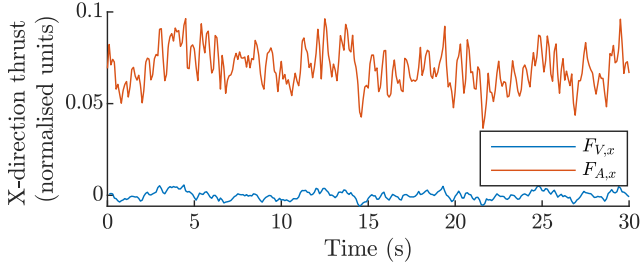


Figure 6: Simulation results of unconstrained MPC, showing the $\mathcal{W}_x$ vectored and attitude-based thrusts used

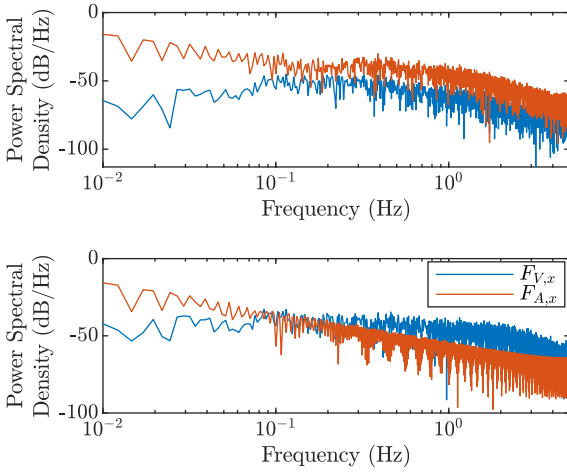


Figure 7: Actuator power spectral densities from simulation. The upper plot shows the frequency content for the default tuning. The lower plot shows the frequency content with more aggressive weights ($w_{F_{V,x}} = 0.666$; $w_{\dot{F}_{A,x}} = 1$)

5.1.2 Experiment

Figure 8 shows the $\mathcal{W}_x$ position trajectory for a 50s period. Overall, the mean is $-8mm$, and the standard deviation is $10mm$. In the other axes, performance is less strong, with the $\mathcal{W}_y$ and $\mathcal{W}_z$ mean values being -82 , and $110mm$ respectively. However, this is easily explained by the fact that wind is only modelled in the $\mathcal{W}_x$ axis. As such disturbances or imbalances in other axes can not be accounted for. The standard deviations for these axes are still low, at $6mm$ and $7mm$ respectively. After removing the steady state errors the RMS position error was found to be $13mm$. This is comparable, to the $\mathcal{H}_\infty$ controller from [5], albeit slightly worse.

5.2 Motor Constraint Testing

The simulation stability of the controllers under different motor constraint conditions is shown in Table 2. Figure 9 then

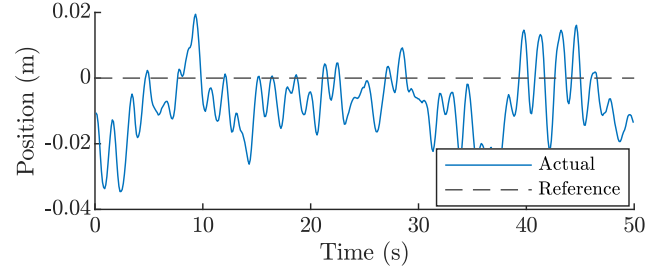


Figure 8: Experimental unconstrained MPC position tracking

shows the position trajectories of the $\mathcal{H}_\infty$ and Model Predictive Controllers for the test case $[0.405 \ 0.595]$. Overall, the $\mathcal{H}_\infty$ controller appears to be slightly better at handling actuator saturation. This is counter-intuitive, but it is theorised to be related to the inherent redundancy provided by the craft's 8 rotors. When 4 rotors are saturated, the aircraft can often still maintain quadrotor-like control schemes. Meanwhile, the fact that the MPC destabilises earlier is suspected to be related to the approximate nature of constraints and the extensive linearisation use. The ultimate conclusion is that this cascade MPC architecture is insufficient for handling actuator limits.

Lower PWM Limit	Upper PWM Limit	$\mathcal{H}_\infty$ Stable	MPC Stable
0.4	0.6	Y	Y
0.405	0.595	Y	N
0.41	0.59	N	N
0.4	1	Y	Y
0.425	1	Y	N
0.430	1	N	N
0	0.54	Y	N
0	0.535	N	N

Table 2: Constrained simulation stability. Full range: $[0 \ 1]$

5.3 Angle Constraint Testing

5.3.1 Simulation

From trim analysis, the required pitch angle for a $5.6m/s$ wind is 8° . As such, the controllers are tested with a smaller limit of 6° . The controller performance is shown in Figure 10, demonstrating that with MPC, the constraint is obeyed, with only occasional violations. Furthermore, the MPC shows far better regulation performance. The minor violations can be compensated for with a safety factor in the constraint.

5.3.2 Experimental Results

When angle is constrained to 6° under a wind of $5.6m/s$, the $\mathcal{W}_x$ axis results can be seen in Figure 11. In this case, performance is worse compared to Figure 8, with a steady state error of approximately $-58mm$. This is because, when under

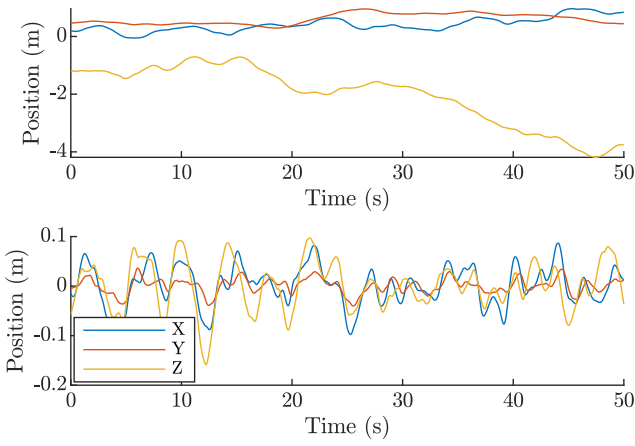


Figure 9: Simulation results of MPC (top) and $\mathcal{H}_\infty$ (bottom) position tracking with imposed constraints [0.405, 0.595]

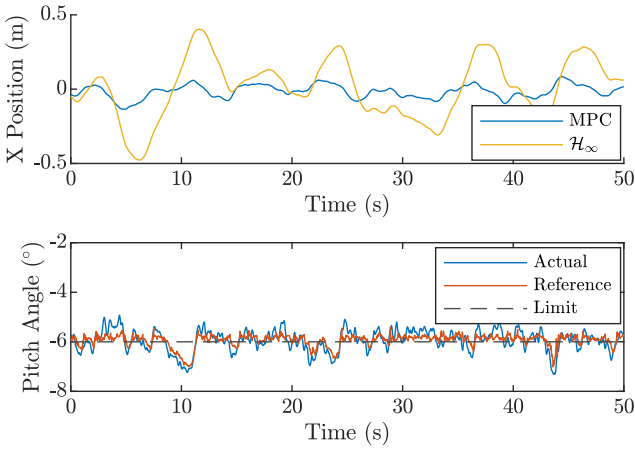


Figure 10: Simulation comparison of MPC and $\mathcal{H}_\infty$ under turbulent wind. Top: $\mathcal{W}_x$ position tracking; Bottom: pitch angle θ with a soft limit of 6° (MPC Only)

an angle constraint, vectored thrust is rejecting mean wind components, while also being weighted at low frequencies. However, this is still less than the performance degradation seen for $\mathcal{H}_\infty$ in simulation. Namely, while simulation and experiment cannot be compared directly, simulation showed that transient angle-constrained $\mathcal{H}_\infty$ performance was much worse than MPC (Figure 10). Translating this relative performance into experiment, it would be expected that the increased variance under $\mathcal{H}_\infty$ exceed the MPC steady-state error. In any case, this steady state error could also be fixed by position-integral weights. The standard deviation in the $\mathcal{W}_x$ axis is also increased compared to the unconstrained case, at 14mm , but the change is slight. Finally, in the $\mathcal{W}_z$ axis, a 377mm steady state error, is introduced under an angle constraint. This is again suspected to be related to mean wind in this axis, and the fact that the UAV is operating at an angle

below that of the linearisation point. As such, the mean vertical thrust required changes, which cannot fully be corrected by MPC, while staying at the origin. As above, this could be fixed with integral weights.

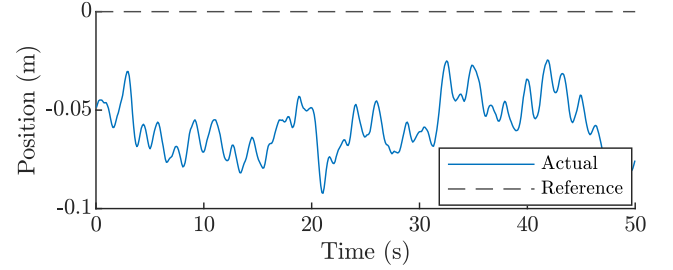


Figure 11: Experimental results of MPC position tracking

The angle constraint was also satisfied as shown in Figure 12. It shows that angle increases up to a limit, after which, ${}^B F_{V,x}$ begins to vary more, and partially reject mean wind.

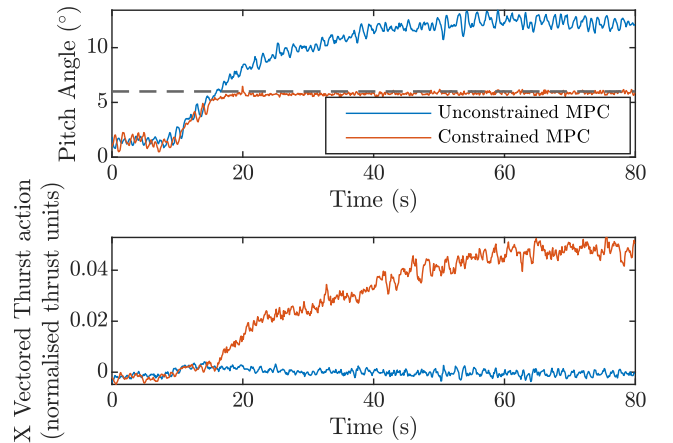


Figure 12: Experimental comparison of constrained and unconstrained MPC with turbulent wind. Top shows pitch angle regulation; bottom shows ${}^B F_{V,x}$ (vectored thrust) usage

6 CONCLUSION

This work has presented the development of a Model Predictive Controller for a fully-actuated UAV under wind disturbances. The ability to handle wind is key for physical environmental interaction, and prior work has established a range of control methods to facilitate this. However, one concern that arose was constraint handling. MPC development for the constraint of motor efforts and orientations has been carried out. Other controller features are weightings that provide frequency-based actuator allocation, and disturbance estimators to handle deviations in the wind from the operating point.

In simulation, UAV motor effort constraints were found to be ineffective compared to the prior $\mathcal{H}_\infty$ controller. However, MPC usage for the constraint of angle showed promise. In

experiments, under an increasing wind, the Model Predictive Controller successfully restrained the angle to 6°.

Future work will more rigorously test the controller. In particular, variations in wind speeds and craft parameters need to be investigated. The controller could also be extended, for example, with angle constraints about more axes.

ACKNOWLEDGEMENTS

The research reported was conducted as part of “Enabling unmanned aerial vehicles (drones) to use tools in complex dynamic environments UOCX2104”, funded by the New Zealand Ministry of Business, Innovation and Employment.

REFERENCES

- [1] Hazim Shakhatreh, Ahmad H. Sawalmeh, Ala Al-Fuqaha, Zuochao Dou, Eyad Almaita, Issa Khalil, Noor Shamsiah Othman, Abdallah Khreishah, and Mohsen Guizani. Unmanned aerial vehicles (UAVs): a survey on civil applications and key research challenges. *IEEE Access*, 7:48572–48634, 2019.
- [2] Victoria Gonzalez-Dugo, P. Zarco-Tejada, E. Nicolás, Pedro Antonio Nortes, J. J. Alarcón, Diego S. Intrigliolo, and E.J.P.A. Fereres. Using high resolution UAV thermal imagery to assess the variability in the water status of five fruit tree species within a commercial orchard. *Precision Agriculture*, 14(6):660–678, 2013.
- [3] Charles T. Howell III, Frank Jones, Taylor Thorson, Richard Grube, Cecil Mellanson, Lee Joyce, John Coggin, and Jack Kennedy. The first government sanctioned delivery of medical supplies by remotely controlled unmanned aerial system (UAS). Technical report, NASA Langley Research Center, 2016.
- [4] James R. Kutia, Karl A. Stol, and Weiliang Xu. Aerial manipulator interactions with trees for canopy sampling. *IEEE/ASME transactions on mechatronics*, 23(4):1740–1749, 2018.
- [5] Jérémie Xavier Joseph Bannwarth. *Aerodynamic modelling and wind disturbance rejection of multirotor unmanned aerial vehicles*. PhD thesis, The University of Auckland, 2021.
- [6] Panfeng Shu, Feng Li, Junjie Zhao, and Masahiro Oya. Robust adaptive control for a novel fully-actuated octocopter uav with wind disturbance. *Journal of Intelligent & Robotic Systems*, 103(6):1–17, 2021.
- [7] Santos Miguel Orozco Soto, Jonathan Cacace, Fabio Ruggiero, and Vincenzo Lippiello. Active disturbance rejection control for the robust flight of a passively tilted hexarotor. *Drones*, 6(9):258, 2022.
- [8] Siddharth Sridhar, Gaurang Gupta, Rumit Kumar, Manish Kumar, and Kelly Cohen. Tilt-rotor quadcopter explored: hardware based dynamics, smart sliding mode controller, attitude hold & wind disturbance scenarios. In *American Control Conference (ACC)*, pages 2005–2010, 2019.
- [9] Shilin Yi, Keigo Watanabe, and Isaku Nagai. Anti-disturbance control of a quadrotor manipulator with tiltable rotors based on integral sliding mode control. *Artificial Life and Robotics*, 26(4):513–522, 2021.
- [10] Gerardo Flores, Andrés Montes de Oca, and Alejandro Flores. Robust nonlinear control for the fully actuated hexa-rotor: theory and experiments. *IEEE Control Systems Letters*, 7:277–282, 2022.
- [11] Pengkang Yu. *An over-actuated multi-rotor aerial platform and iterative learning control applications*. PhD thesis, University of California, Los Angeles, 2022.
- [12] Maximilian Brunner, Karen Bodie, Mina Kamel, Michael Pantic, Weixuan Zhang, Juan Nieto, and Roland Siegwart. Trajectory tracking nonlinear model predictive control for an overactuated MAV. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5342–5348, 2020.
- [13] Davide Bicego, Jacopo Mazzetto, Ruggero Carli, Marcello Farina, and Antonio Franchi. Nonlinear model predictive control with enhanced actuator model for multi-rotor aerial vehicles with generic designs. *Journal of Intelligent & Robotic Systems*, 100(3):1213–1247, 2020.
- [14] Mohit Mehndiratta, Karanjot Singh, Erdal Kayacan, and Mir Feroskhan. Receding horizon-based fault-tolerant control of QuadPlus: an over-actuated quadrotor. In *IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 853–859, 2021.
- [15] Chuanbeibei Shi and Yushu Yu. Design and implementation of a fully-actuated integrated aerial platform based on geometric model predictive control. *Micromachines*, 13(11):1822, 2022.
- [16] Kostas Alexis, George Nikolakopoulos, and Anthony Tzes. Switching model predictive attitude control for a quadrotor helicopter subject to atmospheric disturbances. *Control Engineering Practice*, 19(10):1195–1207, 2011.
- [17] Tong Lv, Yanhua Yang, and Li Chai. Extended state observer based MPC for a quadrotor helicopter subject to wind disturbances. In *Chinese Control Conference (CCC)*, pages 8206–8211, 2019.
- [18] Kostas Alexis, Christos Papachristos, Roland Siegwart, and Anthony Tzes. Robust model predictive flight control of unmanned rotorcrafts. *Journal of Intelligent & Robotic systems*, 81(3-4):443–469, 2015.

Assessing the Agility of a Variable-Tilt UAV

Joshua Taylor,* Matthew Edwards, Karl Stol, Salim Al-zubaidi, and Nicholas Kay

Department of Mechanical and Mechatronics Engineering

University of Auckland, New Zealand

ABSTRACT

Unmanned Aerial Vehicles (UAVs) have grown in popularity over recent years. However, their operation in complex environments with interference from, for example, wind and human interaction demands improved maneuverability and agility. This has brought about the rise of fully and over-actuated UAVs which can decouple translational and rotational dynamics to provide such improvements. This paper focuses on a fully-actuated variable-tilt UAV design with four rotors, all of which can be tilted collectively using two servo motors. A model of the variable-tilt UAV's horizontal thrust dynamics is presented and validated via bench-testing and flight testing of a prototype. Using this model, the key contribution of this paper is a comparison of the variable-tilt concept to an optimized fixed-tilt, over-actuated octocopter in terms of its agility, defined here as the 10-90% rise time of its horizontal thrust force response. While the octocopter currently offers greater agility, with a 70% faster rise time, the potential for improvement of the variable-tilt UAV, as well as other key benefits including improved efficiency, reduced mass, and greater suitability for UAV applications involving interaction, are explored. This provides evidence to justify continued research into variable-tilt UAVs.

1 INTRODUCTION

Unmanned Aerial Vehicles (UAVs) have experienced a rise in popularity in recent times, with applications now ranging from search and rescue to agriculture. Many of these applications require performing complex tasks in dynamic environments, often faced with disturbances such as interaction and wind field fluctuations. Typical under-actuated quadcopters, which have fewer actuators than degrees of freedom (DOFs), are often not suited to these tasks due to their coupled translational and rotational dynamics. Therefore, a need arises for UAVs with improved maneuverability and agility.

This has brought about the rise of fully-actuated and over-actuated UAVs, which have a number of actuators equal to the number of DOFs, or greater than the number of DOFs,

respectively. This allows for the decoupling of DOFs such that the UAV can, for example, travel horizontally without pitching or rolling. The most common type of fully/over-actuated UAV is the fixed-tilt UAV, in which rotors (i.e., the combination of a motor and propeller) are tilted at angles to provide horizontal thrust capabilities. Research in [1] shows how an octocopter with fixed rotors tilted about the body axes improves horizontal thrust bandwidth by an order of magnitude compared to typical under-actuated UAVs. This was improved upon in [2] using a fixed-tilt UAV with a heterogeneous rotor configuration (i.e., 2 large *lift rotors*, and 6 smaller tilted *precision rotors*) which shows a further 54% increase in horizontal thrust bandwidth.

However, since the rotors of these UAVs are tilted at fixed angles and, therefore, produce opposing thrusts, they must compromise on flight endurance. This provides an opportunity to assess the performance of variable-tilt UAVs as an alternative. Variable-tilt UAVs use active tilting of the rotors during flight, typically achieved using servo motors, to provide thrust-vectoring capabilities. Control over the tilt angle allows for more efficient flight as the thrust of individual rotors can be aligned parallel to each other and in the case of hover, can be aligned directly against gravity. Variable-tilt UAVs do, however, suffer from the increased mass of the tilt actuators and the added complexity of design and control. Several variable-tilt UAV designs exist, which can be broadly categorized into independent-tilt and collective-tilt.

In independent-tilt UAVs, the orientation of each rotor can be controlled independently. Designs exist in the literature that allows for tilting in the cant angle (i.e., about an axis in line with the UAV arm) [3, 4, 5], dihedral angle (i.e., about an axis orthogonal to the UAV arm) [6, 7], as well as in both angles [8, 9]. Collective-tilt involves the simultaneous tilting of all rotors, typically via a mechanical linkage, and, therefore, usually allows for tilting in both the cant and dihedral angles [10, 11, 12]. While independent-tilt UAVs provide an element of redundancy, collective-tilting allows for fewer actuators while remaining fully-actuated, thereby benefiting from more straightforward design and control, as well as less energy expended during flight.

Assessment of the ability of these UAVs to operate in complex, dynamic environments requires a metric for agility. Several different methods have been used in the literature. In [13] a comprehensive set of 9 metrics is proposed for determining the agility of a UAV. These are determined by applying a step input during a flight test and assessing the attitude change and acceleration, for example by using the

*Email address(es): jtay996@aucklanduni.ac.nz

bandwidth based on 10–90% rise time. Alternatively, a metric is used in [1, 2], which defines agility as the horizontal thrust force bandwidth. Furthermore, in [14], agility is defined as the maximum thrust that can be produced in any direction, although this metric caters more towards omnidirectional UAVs.

In this work, a variable-tilt UAV design is explored as an alternative to fixed-tilt UAVs for complex tasks in dynamic environments. The key contribution is the use of a model of the UAV to assess its performance in terms of its agility and compare this to an existing fixed-tilt over-actuated octocopter. Another contribution is the build of a functioning prototype for validation of the model used. In Section 2, the chosen variable-tilt UAV design is described, after which the metric used to describe agility in the context of variable-tilt UAVs is covered in Section 3. Section 4 details the numerical model used for assessment of the UAV, followed by validation of the model in Section 5 and a comparison to a fixed-tilt UAV in Section 6.

2 VARIABLE-TILT CONCEPT

The fully-actuated collective-tilt concept presented in [11] is chosen as a variable-tilt UAV to assess. This is shown in Figure 1, where f_{P_i} and m_{P_i} are the thrust force and moment produced by the i th rotor, and body-fixed axes are defined at the center of the upper structure. This design uses a 2-DOF actuated joint at the center of the UAV, which collectively drives four rotors using a mechanism consisting of four four-bar linkages. All joints, excluding the actuated joint, provide three rotational DOFs to allow for tilting of the rotors in any direction.

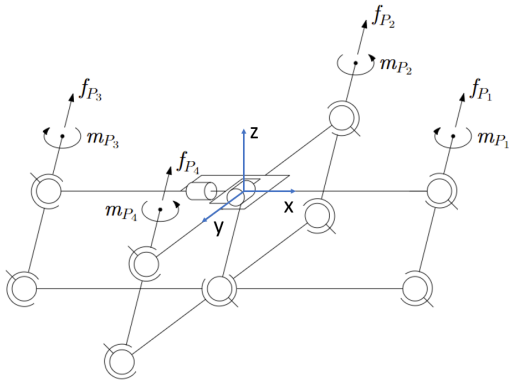


Figure 1: The variable-tilt concept presented in [11] assessed in this work

The coordinate system used to describe rotor orientation is shown in Figure 2, where θ is referred to as the tilt angle, and ϕ as the azimuth. Given that the four-bar linkages that drive the tilting of the rotors form parallelograms, the orientation of each rotor and the center linkage is identical and can be described by one shared tilt angle and azimuth.

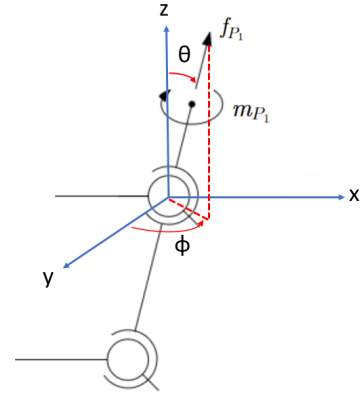


Figure 2: The coordinate system used to describe the orientation of the rotors

3 AGILITY

As described in [13], there are various approaches to measuring agility, but no standard for multi-rotors. The metric chosen in this paper is the 10-90% rise time of horizontal thrust to a step input. This measure allows for non-linearities such as those introduced by the saturation of servo motors typically used for tilting.

More specifically, agility will be measured from the open-loop response of the UAV to a step input that drives a transition from zero horizontal thrust to maximum horizontal thrust. In the case of the variable-tilt UAV, this will involve a step input to the tilt actuators and a simultaneous voltage step input to the rotors in order to maintain vertical equilibrium.

4 NUMERICAL MODEL

A model of the variable-tilt UAVs ability to produce horizontal thrust is presented to measure its response to the step input described in Section 3. The characteristics of the horizontal thrust response are assumed to be dependent purely on the dynamics of the rotors and the tilting actuators; therefore, the model comprises these parts. While geometric coupling between the rotor and servo motor is considered in the model, the effects of dynamic coupling, for example, the effect of the spinning rotor on the tilting actuator load as it passes through the air during tilting, is assumed to be negligible. Additionally, the model assumes that no latency is introduced into the system by electronic components.

The thrust, T , in N produced by each rotor is modeled as a function of its rotational velocity, ω , in rad/s as per:

$$T = c_T \omega^2 \quad (1)$$

where c_T is the thrust constant.

The direction of thrust relative to the body axis (i.e., the tilt angle) is determined by the tilting actuators' response to the step input. Given that servo motors are used, this is considered to be dominated by saturation and the response is,

therefore, modeled by a rate limit, as per:

$$\dot{\theta} = \begin{cases} \dot{\theta}_{ref} & \text{if } \dot{\theta}_{ref} < \dot{\theta}_{max} \\ \dot{\theta}_{max} & \text{if } \dot{\theta}_{ref} \geq \dot{\theta}_{max} \end{cases} \quad (2)$$

where $\dot{\theta}$ is the rotational velocity of the tilt servo in rad/s , $\dot{\theta}_{ref}$ is the reference rotational velocity, and $\dot{\theta}_{max}$ is the rate limit.

To maintain vertical equilibrium during the step response, the following constraint is applied on the UAV throughout:

$$4T \cos(\theta) = mg \quad (3)$$

where m is the mass of the UAV in kg , and g is gravity acceleration in m/s^2 .

Changes in rotor speed are required to satisfy this constraint, as per (1), and the speed dynamics of each rotor are modeled using the non-linear equation of motion presented in [15]:

$$J_{rotor} \dot{\omega}_i = \frac{V_0 - I_0 R}{RK_V V_0} \left(V - I_0 R - \frac{V_0 - I_0 R}{K_V V_0} \omega_i \right) - c_\tau \omega_i^2 \quad (4)$$

where J_{rotor} is the inertia of the rotor in kgm^2 , $\dot{\omega}_i$ is the rotational acceleration of the i th rotor in rad/s^2 , V_0 is the motor's nominal Voltage in V , I_0 is the idle current in mA , R is the motor resistance in Ω , K_V is the motor speed constant in $rad/s/V$, and c_τ is the torque constant in Nms/rad .

5 VALIDATION OF THE MODEL

To validate the model of the variable-tilt UAV's horizontal thrust dynamics, a physical prototype of the variable-tilt concept was built for bench and flight testing. Bench-testing allowed for more accurate sensor data when measuring the force response and was used to validate the model across a range of parameter changes, including changes to flying mass, tilt angle, and azimuth. The flight test validates the bench-test data, as well as the expected flight dynamics of decoupled DOFs.

5.1 Prototype

The prototype developed is shown in Figure 3. The sheet components of the main body and arms are made from cut carbon fiber or acrylic, and the remaining components are primarily designed as 3D-printed parts. The joints are realized using bearings to allow for rotation while still maintaining rigidity in the arms. To assess the capabilities of the UAV, only tilting in on-axis azimuths (i.e., 0° , 90° , 180° , and 270°) is tested, and therefore the use of 2-DOF joints for the four-bar linkages is sufficient. The key components used in the UAV prototype are listed in Table 1.

The model parameters corresponding to the UAV prototype using these components are listed in Table 2. J_{rotor} was determined by modeling the propeller as a slender rigid rod, and c_τ and c_T were determined empirically by fitting curves



Figure 3: The prototype variable-tilt UAV developed for validation

Component	Product
Flight Controller	Pixhawk 4
Motors	RCTIMER MT2610-920KV
Propellers	10x4.7"
Electronic Speed Controllers (ESC)	RCTIMER NFS 30 A
Battery	4S 2500mAh
Tilt Servo	Dynamixel MX-28T

Table 1: Key components used in variable-tilt UAV prototype

to torque-speed and thrust-speed data, respectively, obtained using a load cell. The servo rate limit was obtained experimentally from the saturated region of a step response measured using feedback from a DYNAMIXEL MX-28T servo under no load. Testing under several different loading conditions that might be experienced by the servo motor showed negligible change in the rate limit measured. The remaining model parameters were taken from component specifications provided by the manufacturers.

Parameter	Value
J_{rotor}	5.3763×10^{-5}
c_T	1.976×10^{-5}
c_τ	3.956×10^{-7}
V_0	16 V
K_V	93.64 rad/s/V
I_0	0.7 A
I_{max}	30 A
R	0.11 Ω
$\dot{\theta}_{max}$	5.41 rad/s

Table 2: Model Parameters for UAV Prototype

The characteristics of the prototype variable-tilt UAV are listed in Table 3. Note that the maximum horizontal thrust can be limited by the maximum allowable tilt angle of the frame, or by the maximum thrust that can be produced by the rotors themselves. In the case of this prototype, the limiting factor is the frame, which can tilt up to 15° .

Characteristic	Value
Expected Maximum Horizontal Thrust ($\theta = 15^\circ$)	7.97 N
Payload	1 kg
Expected Hover Endurance	5 min
Flying Mass	3 kg
UAV Radius	300 mm

Table 3: Characteristics of the prototype variable-tilt UAV

5.2 Bench-test

5.2.1 Setup

Bench-testing involved fixing the prototype to a JR3 6DOF force balance as shown in Figure 4. Since the tilting mechanism is located on the underside of the UAV, to allow for a simple connection to the force balance, it was mounted upside-down. As the UAV was raised from the ground, the mounting orientation was considered to have a negligible effect on the rotor aerodynamics. The force balance has a resolution of 0.005 N and an upper limit of 40 N in the horizontal plane, and samples were acquired at 1000 Hz via a data acquisition card. To allow for long periods of testing with consistent voltages, a power supply was used instead of batteries.

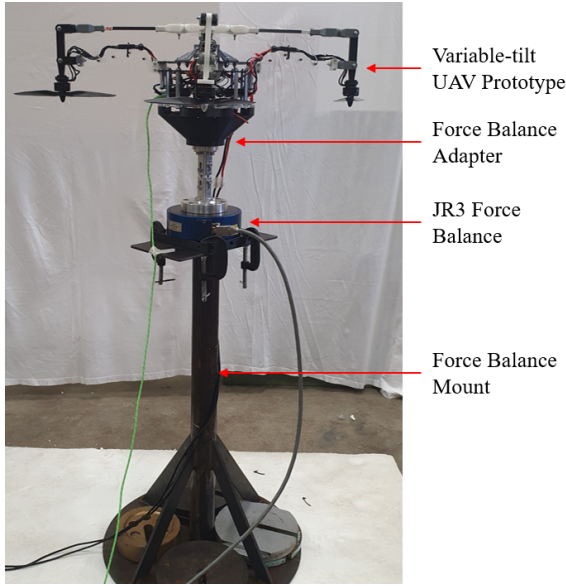


Figure 4: Bench-test setup for the variable-tilt UAV prototype

To apply the required step inputs to measure agility, the pulse width modulation (PWM) signal sent to the ESCs was controlled directly, bypassing any control schemes typically used by the flight controller. Given the flying mass of the UAV that is being tested and the equilibrium condition (3), the hover thrust that needs to be produced via the rotors is known, and using (1) can be used to determine the necessary rotational speed of the rotors. Using (4) at steady-state (i.e.,

$\dot{\omega} = 0$) then relates V to this required ω , thereby defining the reference voltage values before and after a horizontal thrust is produced. The duty cycle, D , of the corresponding PWM signal used to control the rotor voltage can be determined by:

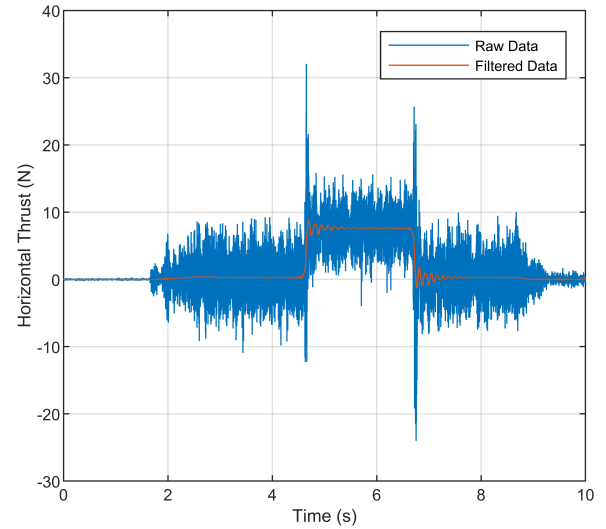
$$V = DV_0 \quad (5)$$

where D then translates to a 1-2ms pulse sent to the ESCs.

5.2.2 Signal Processing

Due to the low signal-to-noise ratio (SNR) of the data collected by the force balance, as can be seen in Figure 5, the data is processed before analysis. A significant portion of this noise can be linked to vibration caused by rotor asymmetry. Testing of individual rotors showed that resonance is reached near the range of hover conditions tested during bench-testing (i.e., 1400-1500 μs) as shown in Figure 6.

Given that filtering has the effect of smoothing a response, it is crucial that an appropriate filter is used, and that the time resolution is maintained while minimizing the noise. A third-order low-pass elliptic filter with a cutoff frequency of 20 Hz is chosen. A forward and backward pass of the filter over the data is performed, thereby effectively increasing the order of the filter to sixth-order and ensuring zero phase distortion is introduced. The filtered data is also shown for the sample presented in Figure 5.

Figure 5: Data measured by the JR3 force balance (flying mass = 3 kg, $\theta = 15^\circ$, $\phi = 0^\circ$)

5.2.3 Results

The first test case corresponds to a flying mass of 3 kg, a final tilt angle of 15° , and an azimuth of 0° , such that an expected final horizontal thrust of 7.97 N is produced. The step response is shown in Figure 7, alongside the corresponding

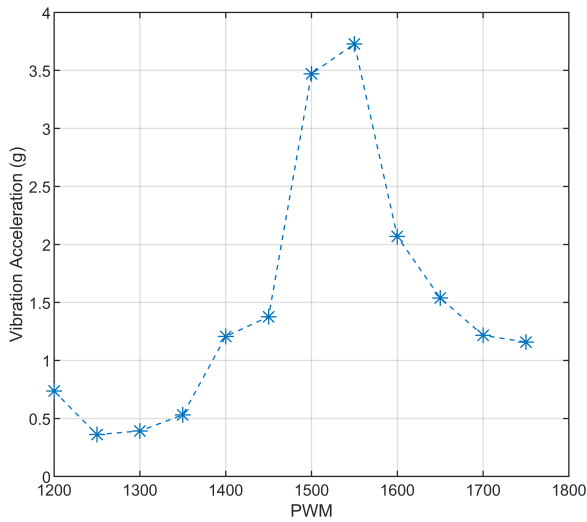


Figure 6: Vibration of the rotor at different PWM signals

response produced by the model. The response shows significant overshoot and oscillation, however, high-speed camera footage of the tilting confirmed that this was a component of the actual response, and not a consequence of filtering. Fortunately, 10-90% rise time as a measure of agility is robust to this, as it does not consider steady-state oscillations to the same extent as a measure such as settling time might.

It is clear from Figure 7 that there is a small amount of error at steady state. This could be the result of imperfections in the frame's geometry leading to inaccurate thrust vectoring or its vibration leading to a reduction in the efficiency of the rotors and, therefore, c_T which was measured for the model by fixing the rotor to a more rigid mount. However, despite the need for improvement in the rotor model, the gradient during saturation is very similar between the model response and the experimental response. Across 8 measured time responses for these testing parameters, a mean 10-90% rise time for the experimental data of 49.7 ms was recorded, which is very similar to the 38.6 ms rise time produced by the model. This provides a foundation for validating the model for the purposes of providing an accurate measure of agility. It is likely that considering the behavior of the servo motor outside the region of saturation would increase the rise time produced by the model, thereby somewhat resolving the small difference between the numerical and experimental data. Additionally, although minimal, residual effects from the filtering are likely to increase experimental rise time slightly, which might further explain the difference to the model.

Given that the rise times measured could vary due to different loading for the two tilt actuators and varying friction in the joints, testing was completed for a tilt angle of 15° along all other on-axis azimuths (i.e., $\phi = 90^\circ$, 180° , and 270°). This was repeated 8 times. The measured rise times are shown in Figure 8, which demonstrates that, while there are

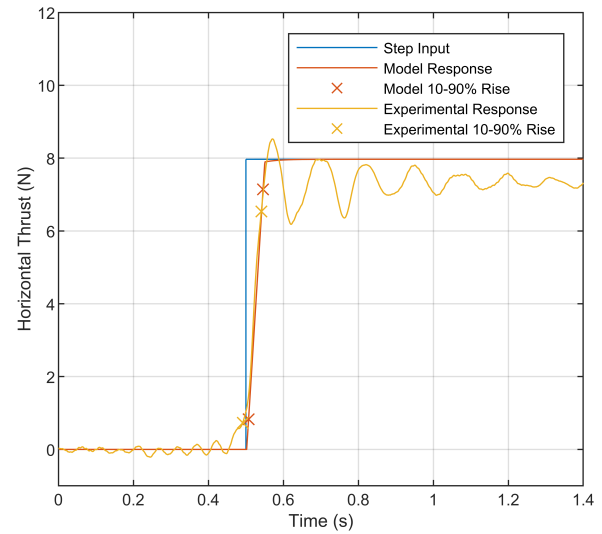


Figure 7: Comparison between the model response and experimental response (flying mass = 3 kg, $\theta = 15^\circ$, $\phi = 0^\circ$)

subtle differences present, for the most part, they remain approximately consistent in all four primary directions, providing further confidence in the model.

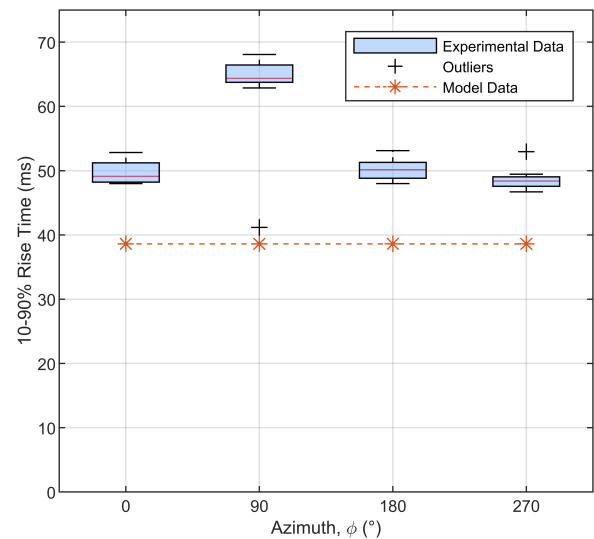


Figure 8: Rise times at different azimuths (flying mass = 3 kg, $\theta = 15^\circ$)

To further validate the model, as well as how well it scales to different parameter changes, bench-testing was conducted for step inputs of a 15° tilt angle at an azimuth of 0° for different hover thrust conditions. PWM values of $1400 \mu s$, $1450 \mu s$, and $1500 \mu s$ were tested which correspond to the hover conditions for flying masses of 2.07 kg, 2.54 kg, and 3.00 kg, respectively. The rise times are shown in Figure 9. Once again, while the rise times are similar, there are slight differences which further suggest that there is a component

of the system that is being misrepresented or ignored in the model. However, the model data appears to accurately capture the trend of the experimental data. This relatively constant trend can also be expected given that the only difference when changing the hover PWM lies in the rotor dynamics, which, as a first-order system, will not change significantly.

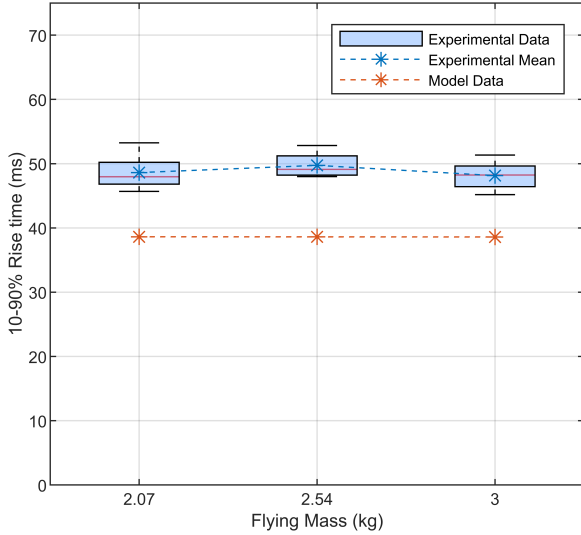


Figure 9: Rise times at different flying masses ($\theta = 15^\circ$, $\phi = 0^\circ$)

Finally, the trend observed when adjusting the tilt angle of the UAV for a 3 kg flying mass and azimuth of 0° was tested and compared, as shown in Figure 10. As with previous testing, there is a slight difference in rise times between the experimental and model data, however, the trend remains approximately the same. In both cases, a rising trend in 10-90% rise time can be seen as the tilt angle increases. Interestingly, as the tilt angle increases, the model data appears to more accurately represent the actual system. This is likely because, for greater tilt angles, there is a larger saturated portion of the response relative to the response as a whole. Therefore, at larger tilt angles, the system more closely resembles the model when dominated by a rate limit.

Overall, while there exist some differences between the 10-90% rise time values produced by the model and experimental data, the model is reasonably validated, in particular for larger tilt angles and for the purposes of analyzing trends produced by varying model input parameters.

5.3 Flight Tests

5.3.1 Setup

A step input was applied to the tilting actuators from 0° to 15° , and a 1 kg payload was mounted to the UAV (i.e., a flying mass of 3 kg), to replicate the test shown in Figure 7. The rotor throttle was adjusted throughout flight to maintain vertical equilibrium, however, this was performed manually and, therefore, the results presented for flight tests are indicative

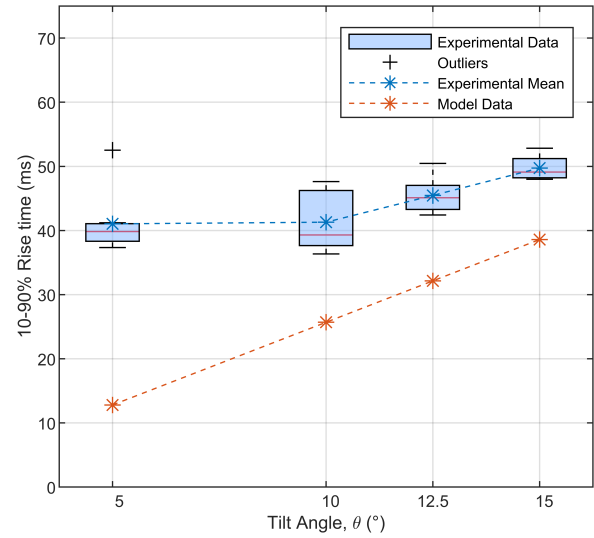


Figure 10: Rise times at different tilt angles (flying mass = 3 kg, $\phi = 0^\circ$)

only. The position of the UAV was recorded during flight with millimeter accuracy using a Vicon motion capture system. The second derivative of this position data gives the acceleration response, of which the measured rise time corresponds to the rise time of the horizontal thrust force response.

5.3.2 Results

Observation of the UAV's flight validates its ability to decouple DOFs. The UAV traveled horizontally due to tilted rotors, while the main body of the UAV remained level with the ground. This is demonstrated in Figure 11, which shows less than 1° of rotation about the pitch axis during horizontal flight. Although it is minor, the UAV pitch angle experienced a sudden drop and recovery as the step input was initially applied, which is likely due to the momentum of the tilting structure, as well as the minor change in center of mass, which had to be accounted for by the pitch controller.

A comparison of the rise time measured from the acceleration response shown in Figure 11 with that measured from the equivalent bench-test validates the accuracy of the data collected from the bench-testing set-up. For a 3 kg flying mass and a final tilt angle of 15° in the 0° azimuth, the bench-test yielded a mean rise time of 49.7 ms, whereas the flight test produced a slightly higher value of 60.1 ms. Since the flight test accounts for all of the UAV's dynamics, instead of isolating the tilting mechanism itself as in the bench-test, there exists a slight difference in these values. Note that this difference is likely magnified by errors introduced by manual control during flight, as the UAV's altitude increased by 233 mm over the time sample shown in Figure 11. However, given that the flight test rise time is in the same order of magnitude as that measured via equivalent bench-testing, this pro-

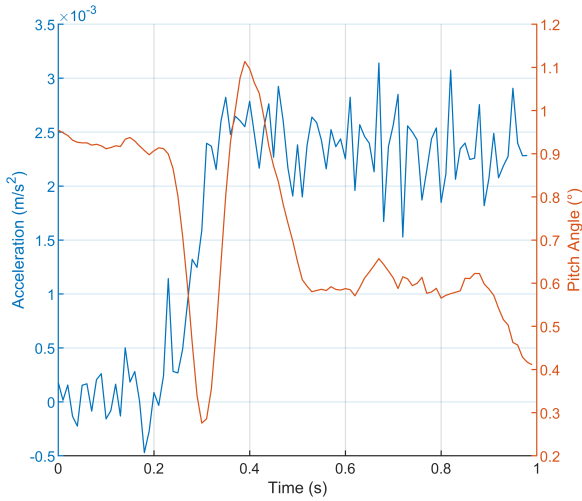


Figure 11: Horizontal acceleration and pitch angle during step-input (flying mass = 3 kg, $\theta = 15^\circ$)

vides further confidence in the bench-test data.

6 COMPARISON TO A FIXED-TILT UAV

To assess how effective the variable-tilt platform is as a method of achieving fully-actuated flight, a comparison is drawn to an existing fixed-tilt, over-actuated octocopter [1, 16]. This UAV is shown in Figure 12 where D_{UAV} is the UAV's diameter, r_{UAV} is the radius, T is the thrust, and β is the common tilt angle about the body (i.e., orthogonal) axes.

In this comparison, the agility of both UAVs is assessed using numerical models. As well as the model of the variable-tilt UAV's horizontal thrust dynamics having been validated, use of this for comparison is deemed acceptable given that the numerical model of the fixed-tilt UAV uses the same equations to describe rotor dynamics [16].

In [16], a parameter sweep is used to generate the parameters and performance characteristics of an optimized octocopter. This draws from a collection of rotors and a range of tilt angles and airframe diameters to find the best octocopter configuration that satisfies three constraints: a specified payload, minimum flight time, and minimum level of horizontal thrust. In this case, these constraints are defined by the variable-tilt UAV's characteristics from Table 3 as 1 kg, 5 min, and 7.97 N, respectively. Table 4 provides a comparison between the resultant octocopter and the variable-tilt UAV.

While the octocopter outperforms the variable-tilt UAV in terms of agility, it is important to consider that the comparison is made between an unoptimized variable-tilt UAV and an optimized fixed-tilt UAV. Since the rise time values between the two UAVs are within the same order of magnitude, this indicates potential for the variable-tilt UAV. A simple parameter sweep that takes in a range of servo motors with higher rate limits could offer improvements to rise time.

Furthermore, it is interesting to note that the variable-tilt

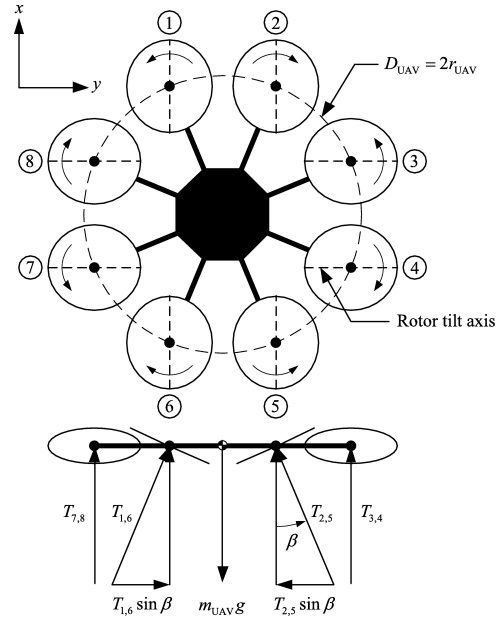


Figure 12: Fixed-tilt over-actuated octocopter [16]

Characteristic	Variable-tilt UAV	Fixed-tilt UAV
Flying Mass	3 kg	3.41 kg
UAV Radius	300 mm	300 mm
Maximum Horizontal Thrust	7.97 N	8.43 N
Flight Time	5 min	5 min
Agility (10-90% rise time)	38.6 ms	11.6 ms
Adjacent Rotor Clearance	170 mm	157 mm

Table 4: Comparison between the variable-tilt design and the fixed-tilt design

UAV achieves similar flight characteristics and performance at a lower flying mass. A key contributing factor here is the battery mass. The required battery mass of the fixed-tilt UAV to achieve the set flight time and payload is 868 g, whereas the variable-tilt UAV only requires a 212 g battery. This is due to the improved flight efficiency, as the rotors can produce direct vertical thrust during hover. This, therefore, addresses one of the key limitations of fixed/over-actuated UAVs discussed in the literature.

Another consideration to be made when comparing the two designs relates to the applications of the UAVs themselves. Given that it is often desirable to fit a manipulator to the front of a UAV, it is beneficial to have greater clearance between adjacent rotor tips, which the variable-tilt UAV has.

7 CONCLUSION

This paper assesses a variable-tilt, fully-actuated quadcopter as an alternative to a fixed-tilt, over-actuated octocopter for use in applications where the UAV must deal with

complex disturbances such as wind-field fluctuations and interaction. Performance is measured in terms of the UAV's agility, defined as the 10-90% rise time of the open-loop horizontal thrust response. A validated model was developed to measure the agility of the variable-tilt UAV and this was used to conduct the comparison to the octocopter. While the rise time is shorter for the fixed-tilt UAV, given that an unoptimized variable-tilt UAV produced results within the same order of magnitude and benefits from improved efficiency, there is reason to continue research in the area of variable-tilt UAVs for applications in dynamic environments.

Future work will involve further improvements to the model such that it can be used in an optimization of the design of the variable-tilt UAV. This will allow for an improved comparison between the variable-tilt and fixed-tilt UAVs. Following this, the study would benefit from further verification via flight testing. To provide horizontal thrust in all directions (i.e., all azimuths), 3-DOF joints will be added to the UAV.

ACKNOWLEDGEMENTS

The research reported in this article was conducted as part of "Enabling unmanned aerial vehicles (UAVs) to use tools in complex dynamic environments UOCX2104", which is funded by the New Zealand Ministry of Business, Innovation and Employment.

REFERENCES

- [1] Z Jeremy Chen, Jérémie XJ Bannwarth, Karl A Stol, and Peter J Richards. Analysis of a multirotor uav with tilted-rotors for the purposes of disturbance rejection. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 864–873. IEEE, 2018.
- [2] Salim Al-Zubaidi and Karl A Stol. Preliminary design optimisation of a novel fixed-tilt heterogeneous uav for horizontal agility. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1489–1496. IEEE, 2022.
- [3] Alireza Nemati and Manish Kumar. Modeling and control of a single axis tilting quadcopter. In *2014 American Control Conference*, pages 3077–3082. IEEE, 2014.
- [4] Markus Ryll, Heinrich H Bühlhoff, and Paolo Robuffo Giordano. A novel overactuated quadrotor unmanned aerial vehicle: Modeling, control, and experimental validation. *IEEE Transactions on Control Systems Technology*, 23(2):540–556, 2014.
- [5] Mina Kamel, Sebastian Verling, Omar Elkhatib, Christian Sprecher, Paula Wulkop, Zachary Taylor, Roland Siegwart, and Igor Gilitschenski. The voliro omniorientational hexacopter: An agile and maneuverable tiltable-rotor aerial vehicle. *IEEE Robotics & Automation Magazine*, 25(4):34–44, 2018.
- [6] Sherif Badr, Omar Mehrez, and AE Kabeel. A novel modification for a quadrotor design. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 702–710. IEEE, 2016.
- [7] Mohammadhosein HajiAbedini, Fatemeh Zargarbashi, Amin Talaeizadeh, Hossein Nejat Pishkenari, and Aria Alasty. Design and implementation of a novel over-actuated quadrotor with variable dihedral angle. In *2021 9th RSI International Conference on Robotics and Mechatronics (ICRoM)*, pages 391–398. IEEE, 2021.
- [8] Ali Bin Junaid, Alejandro Diaz De Cerio Sanchez, Javier Betancor Bosch, Nikolaos Vitzilaos, and Yahya Zweiri. Design and implementation of a dual-axis tilting quadcopter. *Robotics*, 7(4):65, 2018.
- [9] Pau Seguí-Gasco, Yazan Al-Rihani, Hyo-Sang Shin, and Al Savvaris. A novel actuation concept for a multi rotor uav. *Journal of Intelligent & Robotic Systems*, 74(1):173–191, 2014.
- [10] Peter Zheng, Xinkai Tan, Basaran Bahadır Kocer, Er-deng Yang, and Mirko Kovac. Tilt drone: A fully-actuated tilting quadrotor platform. *IEEE Robotics and Automation Letters*, 5(4):6845–6852, 2020.
- [11] Marcin Odelga, Paolo Stegagno, and Heinrich H Bühlhoff. A fully actuated quadrotor uav with a propeller tilting mechanism: Modeling and control. In *2016 IEEE international conference on advanced intelligent mechatronics (AIM)*, pages 306–311. IEEE, 2016.
- [12] Michelangelo Nigro, Francesco Pierri, and Fabrizio Caccavale. Preliminary design, modeling and control of a fully actuated quadrotor uav. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1108–1116. IEEE, 2019.
- [13] Jon Verbeke and Joris De Schutter. Experimental maneuverability and agility quantification for rotary unmanned aerial vehicle. *International Journal of Micro Air Vehicles*, 10(1):3–11, 2018.
- [14] Dario Brescianini and Raffaello D'Andrea. An omnidirectional multirotor vehicle. *Mechatronics*, 55:76–93, 2018.
- [15] Dongjie Shi, Xunhua Dai, Xiaowei Zhang, and Quan Quan. A practical performance evaluation method for electric multicopters. *IEEE/ASME Transactions on Mechatronics*, 22(3):1337–1348, 2017.
- [16] ZJ Chen, KA Stol, and PJ Richards. Preliminary design of multirotor uavs with tilted-rotors for improved disturbance rejection capability. *Aerospace Science and Technology*, 92:635–643, 2019.

Analysis of the Transient Response of UAV Rotors

S. Al-zubaidi* and K. Stol

Department of Mechanical and Mechatronics Engineering, University of Auckland, Auckland

ABSTRACT

This study presents an analysis of the transient response of Unmanned Aerial Vehicle (UAV) rotors, creating models to predict the transient performance of a rotor through manufacturer-provided data. Understanding the transient response helps in choosing rotors for UAVs in a more accurate manner within optimisation, leading to improved agility and stability of UAVs, in addition to improving simulation tools. The models are validated through different experimental measurements, studying the effect of different rotor properties on the performance of the rotor system. These models can be used to estimate the current within 0.3 A and the rise time within 12 ms, which is restricted by the data logging capabilities. The effect of using different Electronics Speed Control (ESC) protocols on the response was also analysed.

1 INTRODUCTION

There has been a notable rise in utilising Unmanned Aerial Vehicles (UAVs) in different areas over the past few years, such as search and rescue [1] and package delivery. Thanks to technological advancements in electronics, UAVs are now more cost-effective and capable of executing intricate operations, especially for multirotor aircraft. Due to their ability to hover in place, take off, and land vertically, they are highly suitable for performing complex manoeuvres and accessing challenging places.

With the increase in interest, there has been a focus on creating UAV designs that are more optimised for certain purposes. These include different aspects of the design to create efficient [2], more dexterous [3], lighter [4], cheaper [5] and high endurance [5, 6] UAVs. With all these optimisations, the performance of the UAV is assessed around its operating hover conditions. Thus, component models that explain the steady-state performance are sufficient. However, with the increased focus on faster responding UAVs [7, 8] studying the performance of the transient response has become more essential. In addition, unlike conventional UAVs, e.g. quadrotors, which require the UAV to pitch/roll to produce horizontal forces, fully and overactuated UAVs [8, 9] can respond to external disturbance much quicker as their response is dependent on the rotor dynamics only. This makes studying transient response or rotor systems very important.

The propulsion system performance of UAVs have been studied extensively and verified through experimental methods in [2, 10, 11, 12], with each component of the system studied from its electrical and mechanical properties. While most of these properties are given by the manufacturer, some of the motor and propeller properties required for these models are not readily available, such as the moment of inertia used in [6].

This problem could be solved through two different pathways. The first pathway is to limit the search space within the optimisation. This way, you can buy and test the motors and propellers' performance and use that as the basis for the optimisation [5, 6, 13]. While this is sufficient to prove that the optimisation is working appropriately, it is not practical to be used to identify the best design from a large database of data such as in [8, 9]. In these cases, the problem is solved by assuming motor inertia is negligible compared to the propeller's. While the optimal UAV result might not change, it would provide an inaccurate representation of the performance. In addition, using this method would not be appropriate if the transient response was a constraint for the optimisation as an infeasible UAV could be chosen.

As for Electronic Speed Controllers (ESC), there have been studies [10] that studied their steady-state effect on performance. Still, they have been largely ignored when studying the effect of their properties on the transient phase. A study has tried to investigate the effect of changing controller firmware on performance, finding improvements in response time [14]. Since then, new protocols such as OneShot, Multishot and DShot have been introduced to improve transient performance and accuracy. However, there have yet to be any studies that quantify the improvement seen with these methods.

The main contribution of this paper is the full analysis of the propulsion unit for its transient response, introducing new models for the moment of inertia of the rotor system components. The aim is to enable the estimation of the transient performance of rotor systems using properties manufacturers provide. In addition, the effect of using different communication protocols for the ESC on the transient response will be analysed.

The remainder of this paper is organised as follows. Section 2 introduces the methodology of testing. The different models used are introduced in Section 3. In Section 4, the results of the testing are shown to verify the models. Section 5 outlines the main findings and future work.

*Email address: salz167@aucklanduni.ac.nz

2 METHODOLOGY

2.1 Test Methods

Different tests have been performed to examine the rotors' performance. The purpose and method of each test are stated below.

2.1.1 Steady-State Test

Within the steady-state test, the rotor is sent a sequence of step inputs and left until the system settles. Data collected is averaged over the last 30 seconds of the response to find the steady-state current, thrust, torque and speed at different levels of ESC inputs.

2.1.2 Step-Response Tests

Within the step-response test, the rotor is sent a sequence of step inputs, figure 1. Data within the transient phase of the response is isolated. This is then used to find the system's response time to different inputs.

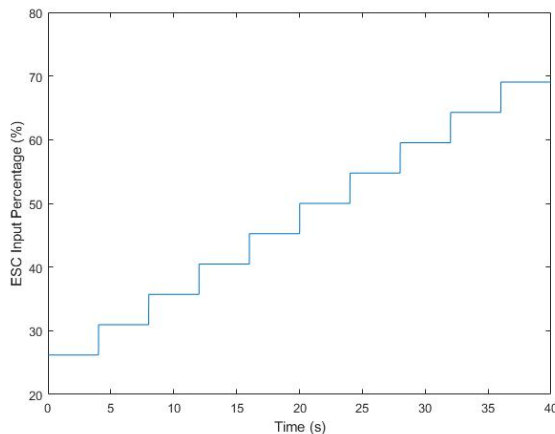


Figure 1: Step-Response Input.

2.2 Data Collection

The response of the propulsion system for different inputs has to be collected. The speed, current, thrust and torque collection were synchronised with the ESC input and collected using the RCbenchmark Dynamometer Series 1580, shown in figure 2. RC Control Board was used to drive the ESC using different protocols. The settings on the Dynamometer were altered to achieve an average sampling rate of 130 Hz. This sampling frequency was high enough to record the transient response. However, the changes in the settings did not improve the torque and thrust sampling rate as a sampling frequency of only 30 Hz was achieved, which is considered insufficient; thus, the thrust and torque must be determined through another method. The RCbenchmark records the torque using load cells placed on the stand. The speed of the motor is recorded using an electrical speed sensor, which involves connecting a probe to one of the wires connecting the ESC to the DC motor.

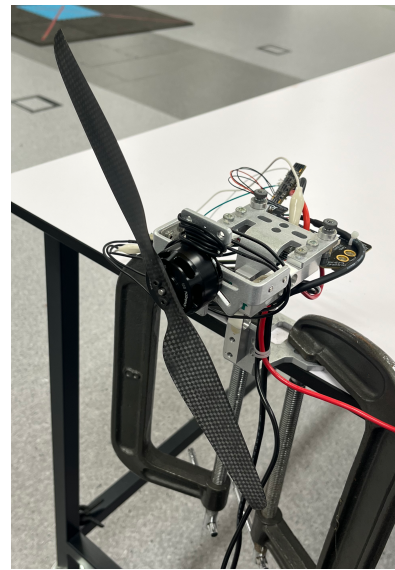


Figure 2: RCbenchmark Test Stand.

3 COMPONENTS

This study defines rotors as the ESC, motor, and propeller systems combined for propulsion within a UAV.

3.1 Propellers

Propellers are responsible for converting the rotational kinetic energy into the thrust required to lift the UAV into the air to hover and perform manoeuvres. The propellers' properties important to consider for this study are torque constant and moment of inertia. Table 1 lists the propellers used in this study.

Propeller	Mass (g)
T-Motor 12" x 4" Carbon Fiber Propeller	14
T-Motor 13" x 4.4" Carbon Fiber Propeller	14.1
T-Motor 14" x 4.8" Carbon Fiber Propeller	15
T-Motor 6" x 2" Carbon Fiber Propeller	1.86
Rctimer TM 6x2 6020 Carbon Fiber Propeller	3
Gemfan F6030 2-Blade Folding Propeller	3.56
HQProp DP 7X4.5 Propeller	6.6
HQProp T5.1X4.5	3.38
HQ T4025	1.45

Table 1: Propellers Used in Case Study.

3.1.1 Torque Constant Model

As the data acquisition rate for torque is lower than that of the other data collected, the torque studied by examining would have to be a theoretical torque. The torque constant relates the rotor load torque with the rotor rotational speed using the common empirical aerodynamic model [9],

$$\tau_l = c_{\tau_l} \omega^2 \quad (1)$$

where τ_l is the rotor load torque, ω is the rotational speed of the motor, and c_{τ_l} is the torque constant. The constant is calculated by fitting equation 1 to the speed and torque data collected from the RCbenchmark steady-state experiments. Once the torque constant is found, the torque is assumed to be a function of the speed for the step-response tests. Alternatively, the torque constant could be calculated from the diameter and pitch using equations from [10].

3.2 Motors

The motor converts electrical energy to mechanical rotational energy required for the propeller to generate thrust. In addition to the capabilities of the motor, this study includes a model for the moment of inertia of the motor according to its geometry and mass. Motors are also tested separately to isolate the system before adding propellers. In addition, the testing was performed with motors with quite a large range of speed constants to ensure that the models used are applicable to an extensive range of UAV Motors. Table 2 lists the motors used in this study

Motor	K_V (rpm/V)	D×H (mm)	Mass (g)
Emax ECO II-2004	3000	24.7×19.2	16.4
Emax ECO II-2004	2000	24.7×19.2	16.6
Emax ECO II-2807	1500	33.9×34	46.9
MN1806	2300	23×23.8	18
MN3110	700	37.7×37.1	80

Table 2: Motors Used in Case Study.

3.2.1 Dynamic Response

For the dynamic response, the model used is the one proposed by Magnussen et al. [14, 6]. As per that model, the equation of motion of the rotor is,

$$J_r \dot{\omega} = \frac{K_T}{R} (V - K_E \omega) - \tau_l \quad (2)$$

where J_r is the rotor moment of inertia, K_T is the motor torque constant, R is the motor internal resistance, K_E is the motor voltage constant, and V is the voltage input. While K_T and K_E are considered equal, the manufacturer does not give them directly but can be derived through the motor speed constant. The equation for this conversion, retrieved from [10], is,

$$K_T = K_E = \frac{V_0 - I_0 R}{K_V V_0} \quad (3)$$

where K_V is the motor speed constant in rad/Vs , V_0 is nominal voltage, and I_0 is the nominal no-load current. Rotor moment of inertia J_r is found through

$$J_r = J_m + J_p \quad (4)$$

where J_m is the motor moment of inertia and J_p is the propeller moment of inertia. The model will be simulated to compare the expected rise time to the experimental data.

3.2.2 Current Model

According to the literature [10], the current through the motor is a function of the voltage, motor parameters and torque. The governing model is

$$I = \frac{V}{V_{in}} \left(\frac{\tau_l K_V V_0}{V_0 - I_0 R} + I_0 \right) \quad (5)$$

where I is the current through the motor and V_{in} is the input voltage.

3.3 ESCs

An ESC is a device that converts the DC voltage from the battery to a three-phase alternating signal required for the brushless DC motors. The effect on the response when using different protocols will be considered. For the rest of the test, *oneshot42* protocol will be used for consistency for the rest of the tests. The ESC used is the Foxeer Reaper F4 128K BL32.

3.3.1 ESC Protocols

The main aim is to understand the effect of different ESC protocols on the response of the rotors. The protocols tested are *PWM50*, *PWM500*, *oneShot125*, *oneShot42*, *multi-shot*, *dshot 150*, and *dshot300*. *Dshot600* and *dshot1200* was not tested due to the capability of the control module.

4 RESULTS

4.1 Propeller Constant Model

The appropriate motors for each propeller were run at different ESC inputs. Following the procedure from section 2.1.1, steady-state data was extracted. For each propeller, equation 1 were fitted which is evaluated to ensure a proper fit. All of the fits were found to have very low root mean square error (RMSE) and high R^2 .

4.2 Moment of Inertia Model

As finding the moment of inertia is quite a complex process, the equations that were suggested within the literature were used to fit the response of the rotor system used. Under this, the assumption was made that the ESC only provides a time delay at the start of the response without having any extra dynamics. This assumption is that the rotor dynamics will only affect the response's rise time. The first-order approximation of the response, from [6], is given by,

$$\frac{\omega}{V} = \frac{K_T}{\tau_r s + 1} \quad (6)$$

where τ_r is the rotor time constant.

$$\tau_r = \frac{J_r R}{K_T^2 + c_{\tau_l} R \omega} \quad (7)$$

The moment of inertia of each rotor pair is found by performing a fit for each input step and comparing it to its

rise time. The relationship between rise time and the time constant is,

$$T_r = 2.2\tau_r \quad (8)$$

where T_r is rise time.

First, the motors were run without a propeller and their moment of inertia was found. Multiple data points for the moment of inertia were found for each motor, and the mean and standard deviation were studied. The data was considered satisfactory as the standard deviation for the moment of inertia was less than 5%. Next, as we aim to find models that depend on data given by the manufacturer, we fit the moment of inertia to the diameter and mass properties. Through that, a fit was found as below,

$$J_m = A_1 + A_2 m_m d_m^2 + A_3 m_m d_m \quad (9)$$

where A_1, A_2, A_3 are constants, m_m is the mass of the motor, and d_m is the motor's outer diameter. The constants are found to be $A_1 = 1.77 \times 10^{-5}$, $A_2 = -0.098$ and $A_3 = 2.71$.

Next, the propellers were added to the system, and the same process was followed to find the moment of inertia of the rotor system. The moment of inertia of the propellers was found using equation 4. The data was considered satisfactory as the standard deviation for the moment of inertia was less than 5%. In addition, we also compared the resulting moment of inertia of the same propeller when using different motors to verify the data quality. Within these, the difference between the individual moment of inertia was within 10% which was considered acceptable for the application. Similar to the motor system, a model for the moment of inertia was used, and the fit found is

$$J_p = B_1 m_p d_p^2 - B_2 m_p d_p \quad (10)$$

where m_p is the mass of the propeller, and d_p is the propeller's diameter. The constants are found to be $B_1 = 0.135$ and $B_2 = 0.012$.

4.3 Dynamic Response

To verify equation 2, the motor was run through different steps of ESC input. The speed and ESC input are recorded. The ESC input is converted to a reference speed using the equation from [10]. A SIMULINK model is created using the equations 2, and the equivalent voltage was used as input for that model. For each, the experimental data was studied against the results from simulation. Figure 3 shows the response for one rotor, specifically the MN1806 using the T-Motor 6" x 2" Carbon Fiber Propeller. As seen, the two responses match each other quite well.

Next, we ran the motor through multiple steps with equal ESC input increments. The process was repeated 10 times, and the rise time was recorded and compared. Figure 4 shows the comparison, and as we can see, the model works quite well for operating speeds up to 1900 rad/s. The rise time

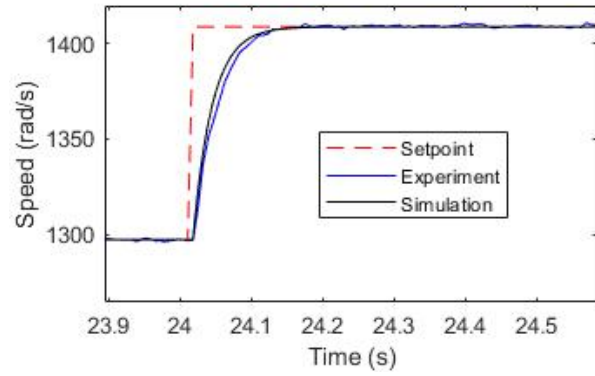


Figure 3: Time Response of One Step

change with operating speed follows a similar trend showing that the models used are quite good. The difference between the experimental and simulated system rise time could be attributed to the limitation in the logging of data. As mentioned above, the average sampling frequency is about 130 Hz, resulting in a sampling period of about 7.7 ms. Looking at the results obtained any over or under-estimation of the rise time was only by about 1-1.5 times that sampling time. While repeating and taking the average reduced the error, small errors are still expected.

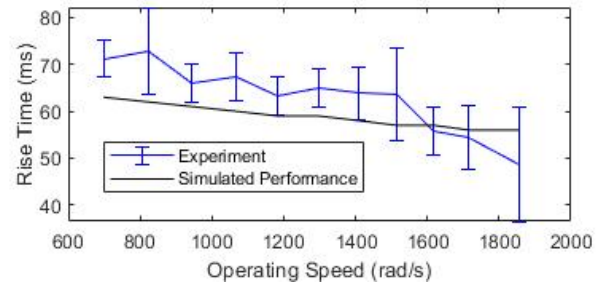


Figure 4: Rise Time at Different Operating Speeds

The same procedure was repeated for the different rotors, and the difference between the expected and actual speed rise times was compared and shown in Table 3 for each trial point. For each rotor, the models used were found to be able to estimate the rise time within less than 1 sampling period with 1 exception. For that rotor system, the rise time was estimated to be within 12.6 ms, translating to only a 6% error.

4.4 Current Response

The appropriate motors for each propeller were run at different ESC inputs. Following the procedure from section 2.1.2, data for each time step has been recorded. The measured current was plotted against the expected current as shown in figure 5 for the Emax ECO II-2807 1500KV motor with the HQProp T5.1X4.5 Propeller. It could be clearly seen

Rotor No.	Absolute Difference (ms)	Percentage Difference (%)
1	5.5	9
2	6.1	8
3	4.2	23
4	6.1	16
5	5	12
6	5.3	14
7	7.8	18
8	3	9
9	12.6	6

Table 3: Absolute and Percentage difference between expected and actual rise time

that the current follows the expected current to a very close estimation (within 5%).

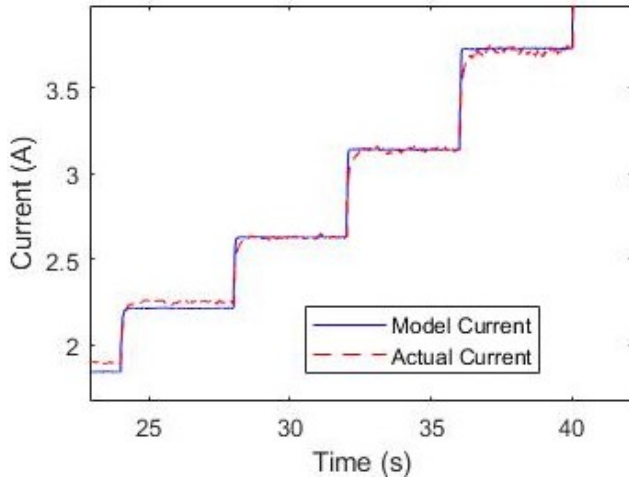


Figure 5: Current Response for Multiple Steps

Secondly, the model was tested with multiple combinations of propeller and motor to verify its performance and validity. For each, the RMSE was calculated between the expected and actual current. As seen in figure 6, 40 trials were conducted of 8 steps, resulting in a maximum RMSE of 0.3 A. This shows that the model provides a good estimation of the current in transient phase.

4.5 ESC Protocols

4.5.1 Dynmaic Response

For the testing of the ESC, we connected the ESC to the MN1806 using the T-Motor 6" x 2" Carbon Fiber Propeller. We tested the system through each of the different input protocols. This was repeated 10 times, and the mean, μ and the standard deviation, σ , of the settling time were recorded. As seen from Table 4, the faster protocols, consistently have the lower settling times, both mean and standard deviation. In

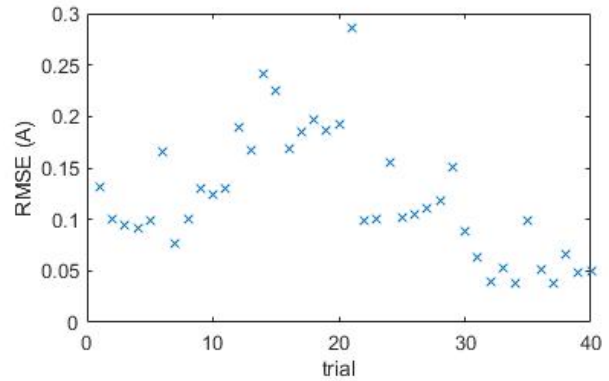


Figure 6: Average current error across trials

Protocol	Step 1		Step 2		Step 3	
	μ	σ	μ	σ	μ	σ
dshot150	101	1.2	98.8	3.5	98.0	3.8
dshot300	105	2.0	103	1.7	99.8	3.1
oneshot42	103	2.9	101	2.2	104	5.3
oneshot125	104	1.6	99.0	8.0	99.4	1.5
multishot	104	1.6	102	4.3	100	6.8
PWM 50Hz	114	4.9	113	7.4	112	7.5
PWM 500Hz	108	2.1	102	4.3	100	9.6

Table 4: Mean and standard deviation of settling time in *ms*

addition, using PWM at either 50Hz or 500Hz causes an increase in the settling time. This is likely caused by the delay due to the longer signal of 1 *ms* - 2 *ms* when using *PWM500* compared to the 125 μ s - 250 μ s for *oneshot125*. However, the difference in settling time between the different protocols is very small, especially considering systems with propellers of size larger than 4". However, this effect might be magnified when working with much smaller propellers.

5 CONCLUSION

In this paper, the transient performance of the actuation system of multirotor UAVs was analysed. Through only data provided by the manufacturer for off-the-shelf components, the rise time for different steps was estimated within 12.6 ms. This provides a good platform to estimate the performance of these components within simulations and optimisations without requiring the purchase and testing of the components.

For future work, the aim is to use these models of transient response within an optimisation, allowing accurate prediction and extensive search space. In addition, expanding the testing range to other limits to explore different problems

ACKNOWLEDGEMENTS

The research reported in this article was conducted as part of "Enabling unmanned aerial vehicles (UAVs) to use tools in complex dynamic environments UOCX2104", which

is funded by the New Zealand Ministry of Business, Innovation and Employment.

REFERENCES

- [1] J. M. Gomez de Gabriel, J. M. Gandarias, F. J. Perez-Maldonado, F. J. Garcia-Nuncz, E. J. Fernandez-Garcia, and A. J. Garcia-Cerezo. Methods for autonomous wristband placement with a search-and-rescue aerial manipulator. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct 2018.
- [2] X. Dai, Q. Quan, J. Ren, and K. Y. Cai. Efficiency optimization and component selection for propulsion systems of electric multicopters. *IEEE transactions on industrial electronics (1982)*, 66(10):7800–7809, Oct 01, 2019.
- [3] D. Brescianini and R. D’Andrea. An omni-directional multirotor vehicle. *Mechatronics (Oxford)*, 55:76–93, Nov 2018.
- [4] P. H. M. Souza and K. Stol. Analysis and design of a novel fully-actuated compact multirotor. In *Australasian Conference on Robotics and Automation (ACRA 2021)*, 2021.
- [5] O. Magnussen, G. Hovland, and M. Ottestad. Multi-copter uav design optimization. In *2014 IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, pages 1–6. IEEE, Sep 2014.
- [6] Ø. Magnussen, M. Ottestad, and G. Hovland. Multi-copter design optimization and validation. *Modeling, Identification and Control*, 36(2):67–79, Jan 01, 2015.
- [7] J. Verbeke, D. Hulens, H. Ramon, T. Goedemé, and J. De Schutter. The design and construction of a high endurance hexacopter suited for narrow corridors. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 543–551, 2014. ID: 1.
- [8] S. Al-Zubaidi and K. A. Stol. Preliminary design optimisation of a novel fixed-tilt heterogeneous uav for horizontal agility. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1489–1496. IEEE, Jun 21, 2022.
- [9] Z. J. Chen, K. A. Stol, and P. J. Richards. Preliminary design of multirotor uavs with tilted-rotors for improved disturbance rejection capability. *Aerospace science and technology*, 92:635–643, Sep 2019.
- [10] D. Shi, X. Dai, X. Zhang, and Q. Quan. A practical performance evaluation method for electric multicopters. *IEEE/ASME transactions on mechatronics*, 22(3):1337–1348, Jun 2017.
- [11] X. Dai, Q. Quan, J. Ren, and K.-Y. Cai. An analytical design-optimization method for electric propulsion systems of multicopter uavs with desired hovering endurance. *IEEE/ASME transactions on mechatronics*, 24(1):228–239, Feb 01, 2019.
- [12] Z. Czyz, P. Karpinski, and K. Skiba. Experimental study of propellers for the electric propulsion system. In *2021 IEEE 8th International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, pages 682–686. IEEE, Jun 23, 2021.
- [13] A. Ayaz and A. Rasheed. Multi-objective design optimization of multicopter using genetic algorithm. In *2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST)*, pages 177–182. IEEE, Jan 12, 2021.
- [14] O. Magnussen, G. Hovland, M. Ottestad, and S. Kirby. Experimental study on the influence of controller firmware on multirotor actuator dynamics. In *2014 IEEE International Symposium on Robot and Sensors Environments (ROSE) Proceedings*, pages 106–111. IEEE, Oct 2014.

Multi-Vehicle Simulation Framework for Heterogeneous Unconventional MAVs

Luiz F. T. Fernandez* ^{1,2}, Murat Bronz* ^{†1}, Thierry Lefebvre^{‡2}, and Nathalie Bartoli^{§2}

¹ENAC, Université de Toulouse, F-31400 Toulouse, France

²ONERA/DTIS, Université de Toulouse, Toulouse, France

ABSTRACT

The growing interest in using aerial vehicles for diverse missions creates the need for higher autonomy and dexterity. Unconventional configurations can potentially satisfy such requirements, at a cost of new challenges in design and control. Simulation environments are often employed to tackle such challenges while minimizing resources expenditure. In this paper, an existing simulation framework has been extended to allow for the modeling of new UAV configurations. Aerodynamic forces and moments generated by propellers and airframes are defined in separate functions to ease the understanding and allow future addition of new vehicle configurations. Incremental nonlinear dynamic inversion and vector field guidance methods are implemented as vehicle agnostic control and guidance solutions. Vehicle models are extracted and implemented into Paparazzi open-source autopilot system, which makes a seamless switch towards real flights and hardware in the loop tests, facilitating control law development. The simulator can also be used within vehicle and mission design environments. With that, the entire closed loop behavior of the system can be addressed from the design stage, allowing for the vehicle and fleet optimization while considering control and operation constraints. Finally, the whole codebase is made available for the community and can be reached at:

github.com/enac-drones/dronesim

1 INTRODUCTION

Simulators have become indispensable tools in aeronautical and robotics research. Their use is one of the factors that led to the increasing interest in deploying aerial robots for

real life missions. In a fully computational environment, researchers can evaluate vehicle performance and different control strategies without expending resources to manufacture and test robots. By employing a simulator to train a learning based solution, Loquercio *et al.* [1] showed how quadcopters can safely and nimbly navigate through complex environments such as forests and disaster zones. Petracek *et al.* [2] also used simulations to assure that two unmanned aerial vehicles (UAVs) could be used to document historical buildings without endangering them. Thanks to the successful application of UAVs in different tests or even in real life scenarios, there is an increasing interest in using them in different tasks. Such missions are increasingly complex, covering demands where time plays a major role, such as health [3] related tasks, and the ones involving more interaction with humans, asking for more dexterity, such as construction [4]. To successfully perform more complex missions, novel vehicle configurations are continually being explored. Winged UAVs [5], capable of higher flight ranges, and different propulsion layouts [6], as the vehicles shown in Figure 1, capable of more dexterous flights, are often preferred. However, they create challenges with respect to accurate physics simulation.

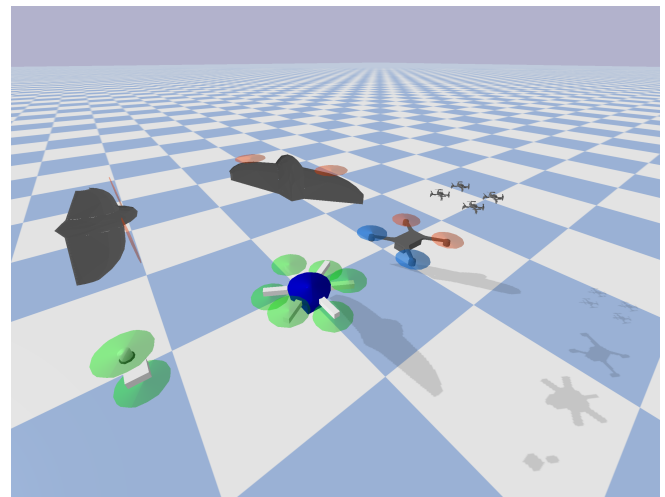


Figure 1: Different vehicle configurations flying together are shown in the simulation environment.

Precise aerodynamic modeling of airframes and propellers then becomes of paramount importance in order to correctly account for vehicle dynamics as it enables the as-

*luiz.tiberio@onera.fr

†murat.bronz@enac.fr

‡thierry.lefebvre@onera.fr

§nathalie.bartoli@onera.fr

*These authors contributed equally to this work

assessment of vehicle stability, control effectiveness, and performance in various flight regimes. Such models are usually built with high order methods in the aeronautical engineering community, using computational fluid dynamics methods as in [7] and [8]. However, the use of such computationally heavy tools remains impracticable at system simulation level for a given trajectory or full mission considering sensing and control systems, as a single point simulation can take hours to run. So, the challenge of accurately addressing aerodynamic behavior of unconventional UAVs within an acceptable computational cost for system simulation remains open.

In addition to the challenges associated with creating new vehicle configurations and defining accurate aerodynamic forces and moments, existing simulators often suffer from limited portability and adaptability. Many simulators are designed to run on specific operating systems or hardware, as NVIDIA GPUs, making them less accessible for researchers using different platforms. This can hinder collaboration and the ability to reproduce research findings across different research groups. In this paper, we present an extension to an existing simulation framework based on Bullet physics engine, which is operating system and hardware-agnostic. The simulator is intended to be used to model new vehicle designs, develop control and guidance laws and simulate mission scenarios while testing different vehicle configurations at the same time. The subsequent sections of this paper delve into the details of related work, proposed solution, and use-case scenarios, highlighting the benefits and potential applications of our approach. Contributions of the paper can be summarized as :

- Facilitate the simulation of new vehicle configurations
- Easier and more intuitive definition of aerodynamic forces and moments
- Heterogeneous multi-vehicle simulation
- Improved portability through different operating systems
- Implementation into a well known open source autopilot system for rapid transition into hardware flights.

2 RELATED WORK

The desire for a simulation tool to evaluate different vehicle designs in [9, 10], and novel control strategies for such vehicles, as in [11], led to the search of available solutions. The interest in testing different guidance strategies [12] and even learning approaches [13] increased even further the need for a reliable and flexible tool. The several available simulators for aerial vehicles are different in terms of objective, implementation, available vehicles, flexibility, and so on. So, the choice of evaluating existing solutions and selecting the most suitable one was made. Fadri *et al.* [14] presented the *RotorS*, a simulator for different multicopter vehicles. The implementation in Robot Operating System (ROS) and Gazebo

facilitates inclusion of sensors for odometry and an Inertial Measurement Unit (IMU). In the same vein, Baca *et al.* [15] presented the *MRS UAV system*, a ROS-based solution to simulate different vehicles while enabling real world deployment and facilitating reproducible research. Yunlong *et al.* [16] presented the *Flighmare*, a flexible quadrotor simulator based on Unity and a physics engine that allows for the inclusion of sensing system on the simulation, evaluation of reinforcement based strategies for control, and more. From the same research group, Foehn *et al.* [17] presented *Agilicious*, an open-source and open-hardware agile quadrotor for vision-based flights. This solution comes with *agilib*, a library with minimal dependencies to integrate *Agilicious* into a given infrastructure, and *agiros*, a ROS based library that allows for easy simulation and real flight. Fernandez-Cortizas *et al.* [18] presented the *Aerostack2*, a ROS2 based simulator for multi-aerial-robots systems. Keipour *et al.* [19] presented a Matlab based simulator for fully-actuated vehicles. With the main focus of studying learning based methods, Kulkarni *et al.* [20] presented the *Aerial Gym Simulator*, based on the NVIDIA Isaac Gym. Panerati *et al.* [21] also presented a Python based simulator adapted for learning methods. The authors introduced an environment called *Gym-Pybullet-Drones*, based on the Bullet physics engine, which supports multiple quadcopters and provides realistic collisions, aerodynamic effects, and reinforcement learning interfaces. With the objective of helping in education, Folk *et al.* [22] presented the *RotorPy*, a python based multirotor simulator that considers aerodynamic effects. More simulation frameworks and research directions can be found in the extensive review by Mairaj *et al.* [23]. Considering the needs of adding different vehicle configurations and aerodynamic and propeller modeling, as well as some aspects of the available simulators, such as maturity, ease of integration with the Paparazzi autopilot system and design environment, we opted to use the *Gym-Pybullet-Drones*, extending it according to our interest.

3 EXTENSIONS

Simulating heterogeneous fleets was one of the main objective for the improvement of the existing framework. However, a unified controller, suitable with most of the new and unconventional vehicle configurations (such as coaxial bi-rotors, fully-actuated hexacopters, and tail-sitters), is also implemented. Some of these vehicles require higher-fidelity models to correctly represent aerodynamic forces and moments. Additionally, for the purpose of mission simulations, a more generic guidance algorithm becomes necessary. All of these issues have been addressed and presented below. This tool is under active development. So new vehicles, control strategies and improvements are added in an ongoing basis.

3.1 Vehicle Agnostic Controller

The controller is based on the work of Van Wijngaarden *et al.* [24] and revolves around the control of the angular accelerations in an *incremental* way. The attitude control law cal-

culates the new command (u_c) as a function of the current command (u_f), current angular acceleration ($\dot{\Omega}_f$) and thrust (T_f) as:

$$u_c = u_f + G^+(\nu - \begin{bmatrix} \dot{\Omega}_f \\ T_f \end{bmatrix}) \quad (1)$$

where G^+ is the Moore–Penrose pseudoinverse of the control effectiveness matrix G and ν is a virtual control law that uses a proportional feedback (k_Ω) to control the angular rates as:

$$\nu = \begin{bmatrix} k_\Omega(\Omega_{ref} - \Omega) \\ T_d \end{bmatrix} \quad (2)$$

where T_d is the desired thrust, calculated by the outer loop, and Ω_{ref} is calculated as a function of the quaternion attitude error (q_{err}), obtained with the Kronecker product between the reference quaternion and the conjugate of the state quaternion:

$$\Omega_{ref} = k_\eta q_{err}^T \quad (3)$$

The velocity control law is:

$$v = v_f + m [G_T(\eta, T) + G_L(\eta, V)]^{-1} (a_{ref} - a_f) \quad (4)$$

where a_{ref} and a_f are the reference and current accelerations respectively, v_f is the current velocity, and $G_T(\eta, T)$ and $G_L(\eta, V)$ are velocity effectiveness matrices for thrust and lift forces, obtained with the derivatives of such forces with respect to the control vector for the outer loop, defined as roll, pitch, and thrust ($[\phi, \theta, T]$). The subscript $[\cdot]_f$ represents the filtered signal for the current acceleration and velocity vectors. The Bullet physics engine runs at 240 Hz, while the controls run at 96 Hz.

3.2 Vector Field Guidance

A generic guidance law becomes beneficial when a wide variety of vehicle configurations needs to be simulated. So we implemented the Vector Field Guidance method from De Marina *et al.* [25]. The basic guidance law uses the velocity error to generate the reference acceleration as:

$$a_{ref} = [v_{ref} - v] k_v \quad (5)$$

where, v is the current velocity, k_v is decomposed as speed and heading gains for tangential and radial directions, and v_{ref} is the reference velocity coming from a predefined vector field. The acceleration terms are neglected, and only velocity information is used. The implementation requires an analytical formulation of the trajectory that is used as a reference, e.g., a circle $x^2 + y^2 + r^2 = 0$. At any point in the environment, a "level-set" can be calculated which has a non-Euclidean distance metric with respect to the reference trajectory, and by driving this distance error towards zero, the vehicle converges to the reference trajectory. Figure 2 shows an example of the outcome of the vector field guidance. The desired velocity (direction and module) for a vehicle at each point in the space (represented by the arrows) is calculated in such a way that the desired trajectory, represented by the circle with radius 100 m, is correctly tracked.

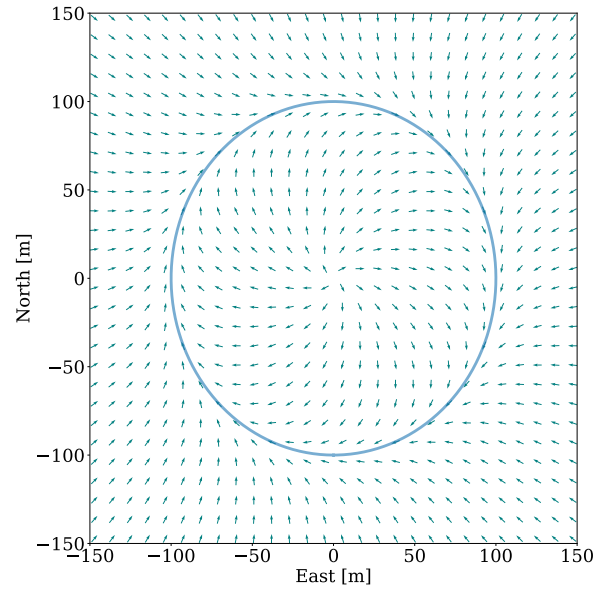


Figure 2: Illustration of vector field guidance, the arrows indicate the velocity direction at each point in the plane.

3.3 Propeller Aerodynamics

To calculate in real time the thrust and moments generated by propellers assuming different geometries and flight conditions, including forward flight, four methodologies are implemented. Both low order data-driven methodologies proposed by Gill and D'Andrea [26] are implemented and indicated when experimental data for thrust and moments for the specific propeller are known. The hybrid blade element momentum (HBEM) by Davoudi and Duraisamy [27] is implemented to account for cases where no performance data is known, only the propeller geometry. In order to use the latter, a valid Matlab license is needed. Lastly, surrogate models can also be used to account for higher fidelity models, enabling a compromise between computational cost and accuracy. Regardless of the surrogate technique or generation strategy, any *Pickle* file containing the model can be read and used for the calculation. Surrogate models created using *CCBlade* [28] were used alongside the simulator in [29] to simulate and optimize a tail-sitter in cruise flight. In [30] seven different methodologies for propeller performance calculation were compared using the proposed simulator. The goal was to observe if, by changing the calculation method in a quadcopter trajectory simulation, the impact in the control input would be noticeable. Despite the fact that desired trajectory was correctly tracked by the vehicle independently of the propulsion methodology, we obtained a significant difference in motor rpm commands, as shown in Figure 3, that compares both methods by Gill and D'Andrea [26] and HBEM [27]. The HBEM led to smaller rpm values, which could lead to a smaller battery requirement during the design phase of a vehicle.

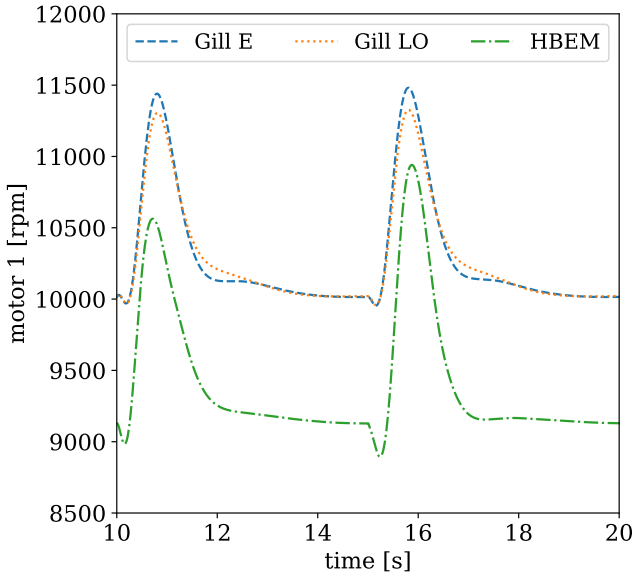


Figure 3: Discrepancies found in rpm commands for a quadcopter for different propeller methods: HBEM and the ones by Gill [30].

3.4 Vertical takeoff and landing UAVs

Modeling the aerodynamic forces and moments experienced by vertical takeoff and landing (VTOL) UAVs remains an open challenge. This kind of vehicle flies in a larger flight envelope than general aviation aircraft, including post-stall conditions. To account for that, the Φ -theory based model, as proposed by Lustosa *et al.* [31] was implemented. This model is algebraically simple and does not rely on angles of attack and sideslip, thus becoming singularity-free, while taking into account the influence of the wing, the deflection of wing control surfaces, and propeller-wing interaction. The same model has already being used by Tal *et al.* [32] to develop a control law capable of enabling agile flight of tail-sitters. In order to make this module applicable for any flying wing configuration, the contributions generated by the propeller-wing interaction in the forces and moments are not taken into account and left for future work, as the original formulation is specific for vehicles with two propellers. Below, we provide a brief explanation of how the aerodynamic forces and moments are generated. For further information, we refer the reader to [31]. Considering a flying wing UAV with 2 elevons and assuming the use of thin airfoils, three Φ -coefficients are needed to build the model using the Φ -theory: the velocity-to-force coefficient $\Phi^{(fv)}$, the velocity-to-moment coefficient $\Phi^{(mv)}$, and the angular-velocity-to-moment coefficient $\Phi^{(m\omega)}$:

$$\Phi^{(fv)} = \begin{bmatrix} C_{D_0} & 0 & 0 \\ 0 & C_{Y_0} & 0 \\ 0 & 0 & 2\pi + C_{D_0} \end{bmatrix} \quad (6)$$

where C_{D_0} and C_{Y_0} are the minimum drag and side force coefficients, respectively.

$$\Phi^{(mv)} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -c^{-1}\Delta_r(2\pi + C_{D_0}) \\ 0 & b^{-1}\Delta_r C_{Y_0} & 0 \end{bmatrix} \quad (7)$$

where b , c , and Δ_r are the wing span, wing mean aerodynamic chord, and flap chord percentage, respectively.

$$\Phi^{(m\omega)} = \begin{bmatrix} C_{l_p} & C_{l_q} & C_{l_r} \\ C_{m_p} & C_{m_q} & C_{m_r} \\ C_{n_p} & C_{n_q} & C_{n_r} \end{bmatrix} \quad (8)$$

where each coefficient accounts for the variation of the rolling (l), pitching (m), or yawing (n) moment coefficient with dimensionless rate of change of roll p , pitch q , or yaw r rate [33]. Additionally, it is necessary to model the so-called elevon effectiveness for both aileron and elevator configurations ζ_f and $\zeta_m \in \mathbb{R}^3$. With that, the aerodynamic force on the body frame (F_b) at the aerodynamic center is calculated as:

$$\begin{aligned} \sum F_b = & -\frac{1}{2}\rho S \Phi^{(fv)} v \mathbf{v}_b + \\ & \frac{1}{4}\rho S \Phi^{(fv)} [\zeta_f \times] (\delta_1 + \delta_2) v \mathbf{v}_b - \\ & \frac{1}{2}\rho S \Phi^{(mv)} v \mathbf{B} \boldsymbol{\omega}_b + \\ & \frac{1}{4}\rho S \Phi^{(mv)} [\zeta_m \times] (\delta_1 + \delta_2) \mathbf{B} v \mathbf{v}_b \end{aligned} \quad (9)$$

where $[\zeta_f \times]$ represents the vector product operation matrix of ζ_f , ρ is the air density, S the wing area, δ_1 and δ_2 are the deflections of each elevon, v is the airspeed, $\mathbf{v}_b$ and $\boldsymbol{\omega}_b$ are linear and angular speed in the body frame, respectively, and $\mathbf{B}$ a matrix defined as a function of wing span (b) and mean aerodynamic chord (c) as:

$$\mathbf{B} = \begin{bmatrix} b & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & b \end{bmatrix} \quad (10)$$

The aerodynamic moment (M_b) acting on the body frame at the aerodynamic center is calculated as:

$$\begin{aligned} \sum M_b = & -\frac{1}{2}\rho S \mathbf{B} \Phi^{(mv)} v \mathbf{v}_b - \\ & \frac{1}{2}\rho S \mathbf{B} \Phi^{(m\omega)} v \boldsymbol{\omega}_b + \\ & \frac{1}{4}\rho S \mathbf{B} \Phi^{(mv)} [\zeta_m \times] (\delta_1 + \delta_2) v \mathbf{v}_b + \\ & \frac{1}{4}\rho S \mathbf{B} \Phi^{(m\omega)} [\zeta_m \times] (\delta_1 + \delta_2) v \mathbf{B} \boldsymbol{\omega}_b \end{aligned} \quad (11)$$

Figure 4 shows a visual comparison between a real and a simulated UAV.

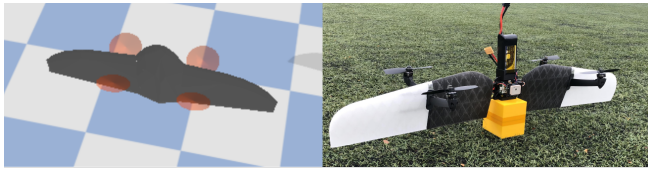


Figure 4: Simulated and real vehicle.

3.5 Adding new extensions

Within the simulation environment, every robot is defined in a Unified Robot Description Format (URDF) file. So, defining a new vehicle and its “physics” consists in creating a new URDF file and defining a new function in the *base aviary*. In this new function, each force and moment acting on the vehicle has to be declared, as well as where (or which link) it is applied. Every existing physics can also be updated to account for new methods, such as a new methodology for propeller performance. Control and guidance strategies are implemented in the *control aviary*, so new solutions should be added to this Python script.

4 USE-CASE SCENARIOS

In the preceding sections, we have outlined the essential building blocks of the framework necessary for simulating a comprehensive use-case. In this section, we present potential applications for the tool. The primary aim is to demonstrate the framework’s potential and provide guidance and motivation for other researchers by offering ready-to-use templates to initiate their own investigations.

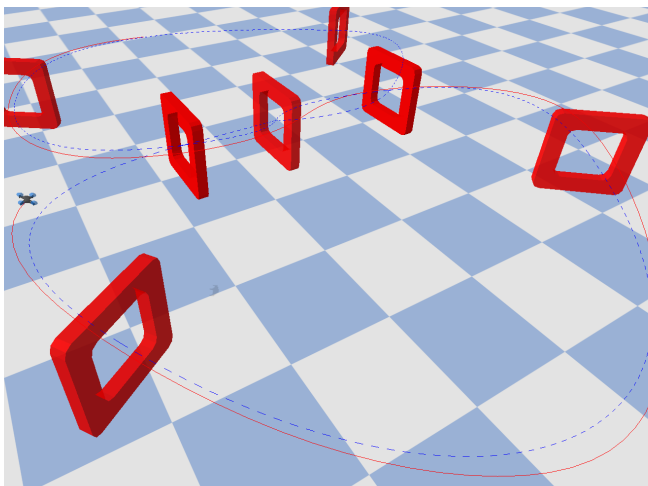


Figure 5: Simulation of a drone passing through gates, representing an example drone racing scenario, with the desired trajectory in blue and the simulated one in red.

4.1 Drone Racing : Time Optimal Trajectory Tracking

Drone racing has emerged as an exciting and rapidly growing sport, captivating both enthusiasts and researchers alike. With its unique blend of speed, agility, and precision, drone racing poses unique challenges that require advanced

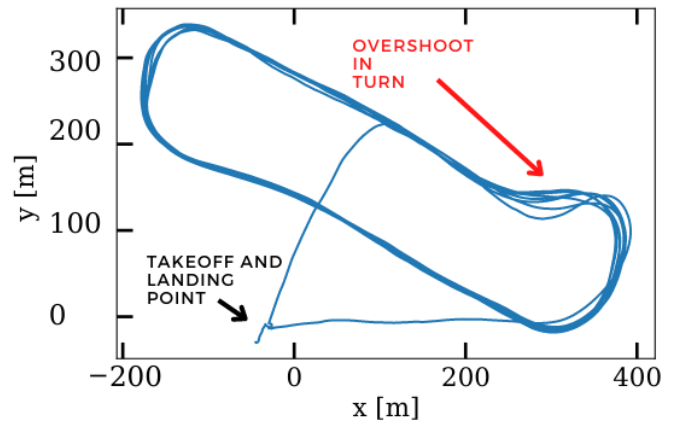


Figure 6: Flight trajectory in the IMAV2022 competition: overshoot in one of the turns caused by strong winds.

control strategies and trajectory tracking techniques. In the example shown in Figure 5, seven gates have been positioned in order to create a figure-eight maneuver. The drone starts from the middle section of the track, and calculates the fastest trajectory through the gates with a predefined sequence. The trajectory is generated by minimizing the snap of the polynomial trajectory, passing through the center of the gates. It can be seen that the desired trajectory calculated by the minimum snap (blue), and the actual flown trajectory (red) has a significant discrepancy. Other planning and control strategies can be implemented and evaluated in this context before real flights.

4.2 Vehicle Design and Control Law Optimization

The proposed simulation framework was used in a multi-disciplinary optimization process applied to a tail-sitter vehicle. A UAV designed in [10] for the IMAV2022¹ presented an undesired overshoot under high wind conditions, as shown in Figure 6. So, in [29], the entire vehicle simulation was embedded in an optimization problem, where the energy consumption for a circular lap was minimized, while reinforcing its wind robustness via constraints. The propulsion layout of the vehicle and the guidance control law gains were the design variables for the problem. With the process, a more energy efficient and robust to wind vehicle was obtained. Figure 7 shows the differences in the flight path of the baseline and optimized vehicle.

5 IMPLEMENTATION TO PAPARAZZI AUTO PILOT SYSTEM

The integration of a simulator environment with a real drone autopilot system, such as Paparazzi[34], offers numerous advantages for the development and verification of control and guidance algorithms. By enabling the seamless trans-

¹<https://www.imavs.org/2022>

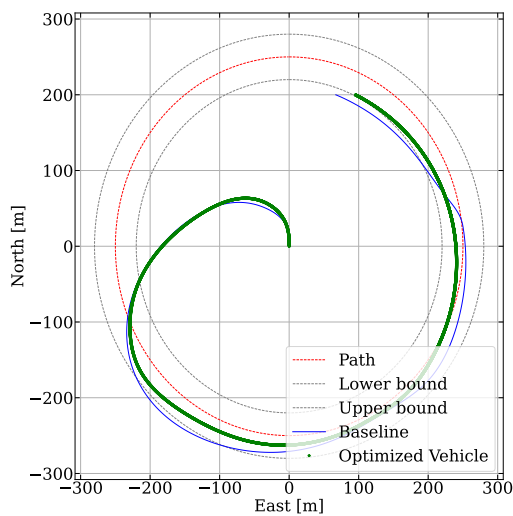


Figure 7: Flight trajectories of baseline and optimized vehicles under wind conditions, with error bounds defined as $\pm 30[m]$.

fer of code between simulation and real-world environments, this approach provides a powerful tool for iterative development, testing, and validation processes. The current develop-

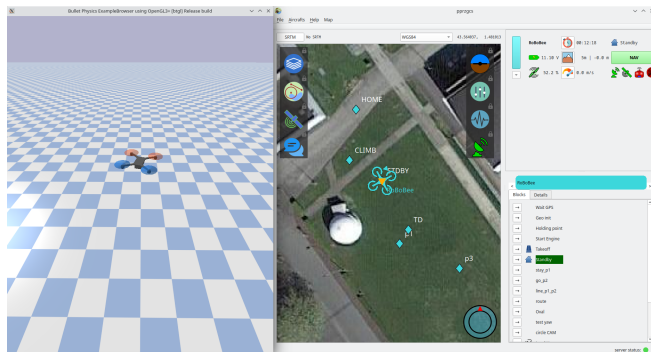


Figure 8: Simulation model running by Paparazzi Autopilot system.

ment state allows for the use of Paparazzi's ground control station to simulate an entire mission. The bridge between simulation and Paparazzi enables the full replacement of the real system with the simulation, while assuming perfect sensing, for the moment. By harnessing the power of simulation, advancements in drone technology can be accelerated, leading to improved aerial capabilities and expanded applications across various industries.

6 CONCLUSION

We presented the extension of an existing framework for simulating aerial robots adapted to account for different vehicle configurations, control and guidance strategies. This open-source solution can be used during design phase, evaluating unconventional vehicle configurations and their perfor-

mance with respect to a given mission. It can also be used for control and guidance laws development, testing, and benchmark. Its simple structure and the usage of Python allows for easy implementation of new features. This tool has already been used for different applications, such as drone racing and tail-sitter design and optimization. Ultimately, it will be used as well for simultaneous mission and heterogeneous swarm of vehicles design.

ACKNOWLEDGEMENTS

This work is part of the activities of ONERA-ISA-ENAC joint research group. The authors would like to thank the ENAC Research Engineer Fabien Bonneval for writing the Paparazzi-pyBullet wrapper that is used in this work.

REFERENCES

- [1] Antonio Loquercio, Elia Kaufmann, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Learning high-speed flight in the wild. *Science Robotics*, 6(59):eabg5810, 2021.
- [2] Pavel Petracek, Vit Kratky, Tomas Baca, Matej Petrlik, and Martin Saska. New era in cultural heritage preservation: Cooperative aerial autonomy for fast digitalization of difficult-to-access interiors of historical monuments. *IEEE Robotics & Automation Magazine*, pages 2–19, 2023.
- [3] Bradley Hiebert, Elysée Nouvet, Vyshnave Jeyabalan, and Lorie Donelle. The application of drones in health-care and health-related services in north america: A scoping review. *Drones*, 4(3):30, Jul 2020.
- [4] Hee-Wook Choi, Hyung-Jin Kim, Sung-Keun Kim, and Wongi S. Na. An overview of drone applications in the construction industry. *Drones*, 7(8):515, Aug 2023.
- [5] Charalampos Vourtsis, Victor Casas Rochel, Nathan Samuel Müller, William Stewart, and Dario Floreano. Wind defiant morphing drones. *Advanced Intelligent Systems*, 5(3):2200297, 2023.
- [6] Alessandro Mancinelli, Ewoud J.J. Smeur, Bart Remes, and Guido de Croon. Dual-axis tilting rotor quad-plane design, simulation, flight and performance comparison with a conventional quad-plane design. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 197–206, 2022.
- [7] Heyecan Koyuncuoglu and Ping He. Cfd based multi-component aerodynamic optimization for wing propeller coupling. In *AIAA SCITECH 2023 Forum*, 2023.
- [8] Eduardo J. Alvarez and Andrew Ning. High-fidelity modeling of multirotor aerodynamic interactions for aircraft design. *AIAA Journal*, 58(10):4385–4400, 2020.

- [9] Elgiz Baskaya, Mahmoud Hamandi, Murat Bronz, and Antonio Franchi. A novel robust hexarotor capable of static hovering in presence of propeller failure. *IEEE Robotics and Automation Letters*, 6(2):4001–4008, 2021.
- [10] Luiz F. T. Fernandez, Murat Bronz, Nathalie Bartoli, and Thierry Lefebvre. Development of a mission-tailored tail-sitter mav. In *13th International Micro Air Vehicle Conference*, pages 159–168, Delft, the Netherlands, Sep 2022. Paper no. IMAV2022-19.
- [11] Jacson MO Barth, Jean-Philippe Condomines, Murat Bronz, Jean-Marc Moschetta, Cédric Join, and Michel Fliess. Model-free control algorithms for micro air vehicles with transitioning flight capabilities. *International Journal of Micro Air Vehicles*, 12:1756829320914264, 2020.
- [12] Zeynep Bilgin, Murat Bronz, and Ilkay Yavrucuk. Panel Method Based Guidance for Fixed Wing Micro Aerial Vehicles. In *International Micro Air Vehicle Conference*, Delft, Netherlands, September 2022. TU-Delft.
- [13] Kirk Hovell, Steve Ulrich, and Murat Bronz. Learned multiagent real-time guidance with applications to quadrotor runway inspection. *Field Robotics*, 2:1105–1133, 03 2022.
- [14] Fadri Furrer, Michael Burri, Markus Achtelik, and Roland Siegwart. *Robot Operating System (ROS): The Complete Reference (Volume 1)*, chapter RotorS—A Modular Gazebo MAV Simulator Framework, pages 595–625. Springer International Publishing, Cham, 2016.
- [15] Tomas Baca, Matej Petrlik, Matous Vrba, Vojtech Spurny, Robert Penicka, Daniel Hert, and Martin Saska. The mrs uav system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles. *Journal of Intelligent & Robotic Systems*, 102, Apr 2021.
- [16] Yunlong Song, Selim Naji, Elia Kaufmann, Antonio Loquercio, and Davide Scaramuzza. Flightmare: A flexible quadrotor simulator. In *Proceedings of the 2020 Conference on Robot Learning*, pages 1147–1157, 2021.
- [17] Philipp Foehn, Elia Kaufmann, Angel Romero, Robert Penicka, Sihao Sun, Leonard Bauersfeld, Thomas Laengle, Giovanni Cioffi, Yunlong Song, Antonio Loquercio, and Davide Scaramuzza. Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight. *AAAS Science Robotics*, 2022.
- [18] Miguel Fernandez-Cortizas, Martin Molina, Pedro Arias-Perez, Rafael Perez-Segui, David Perez-Saura, and Pascual Campoy. Aerostack2: A software framework for developing multi-robot aerial systems, 2023.
- [19] Azarakhsh Keipour, Mohammadreza Mousaei, and Sebastian Scherer. A simulator for fully-actuated uavs, 2023.
- [20] Mihir Kulkarni, Theodor J. L. Forgaard, and Kostas Alexis. Aerial gym – isaac gym simulator for aerial robots, 2023.
- [21] Jacopo Panerati, Hehui Zheng, SiQi Zhou, James Xu, Amanda Prorok, and Angela P. Schoellig. Learning to fly—a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.
- [22] Spencer Folk, James Paulos, and Vijay Kumar. Rotorpy: A python-based multirotor simulator with aerodynamics for education and research, 2023.
- [23] Aakif Mairaj, Asif I. Baba, and Ahmad Y. Javaid. Application specific drone simulators: Recent advances and challenges. *Simulation Modelling Practice and Theory*, 94:100–117, 2019.
- [24] D.C. van Wijngaarden, E.J.J. Smeur, and B.D.W. Remes. Flight code convergence: fixedwing, rotorcraft, hybrid. In *12th International Micro Air Vehicle Conference*, pages 21–27, PUEBLA, MEXICO, Nov 2021.
- [25] 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.
- [26] Rajan Gill and Raffaello D’Andrea. Computationally efficient force and moment models for propellers in uav forward flight applications. *Drones*, 3(4), 2019.
- [27] Behdad Davoudi and Karthikeyan Duraisamy. A hybrid blade element momentum model for flight simulation of rotary wing unmanned aerial vehicles. In *AIAA Aviation 2019 Forum*, 2019.
- [28] Andrew Ning. Using blade element momentum methods with gradient-based design optimization. *Structural and Multidisciplinary Optimization*, 64(2):991–1014, August 2021.
- [29] Luiz F. Fernandez, Murat Bronz, Thierry Lefebvre, and Nathalie Bartoli. Assessment of closed loop dynamics in the multidisciplinary design and optimization of small uavs. In *AIAA AVIATION 2023 Forum*, 2023.

- [30] Luiz F. Fernandez, Murat Bronz, Nathalie Bartoli, and Thierry Lefebvre. Assessment of methods for propeller performance calculation at high incidence angles. In *AIAA SCITECH 2023 Forum*, 2023.
- [31] Leandro R. Lustosa, François Defaÿ, and Jean-Marc Moschetta. Global singularity-free aerodynamic model for algorithmic flight control of tail sitters. *Journal of Guidance, Control, and Dynamics*, 42(2):303–316, 2019.
- [32] Ezra Tal and Sertac Karaman. Global incremental flight control for agile maneuvering of a tailsitter flying wing. *Journal of Guidance, Control, and Dynamics*, 45(12):2332–2349, 2022.
- [33] J. Roskam. *Airplane Flight Dynamics and Automatic Flight Controls*. Number p. 1 in *Airplane Flight Dynamics and Automatic Flight Controls*. Design Analysis and Research Corporation, 1998.
- [34] 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.

Experimentally Validated Simulator of Flight Dynamics: for a Magnus Effect-based Quadcopter System

Zakeye AZAKI*, Pierre SUSBIELLE, Alexis OFFERMANN, Jonathan DUMON,
Nacim MESLEM, Ahmad HABLY, Amaury NEGRE
Univ. Grenoble Alpes, CNRS, Grenoble-INP, GIPSA-lab, F-38000, Grenoble, France

ABSTRACT

Unmanned aerial vehicles (UAVs) have gained extensive utilization across diverse industries, necessitating the enhancement of their capabilities through addressing their power consumption limitations. In this context, the Magnus effect can increase UAV autonomy by exploiting its aerodynamic capabilities. The presented study contributes a reliable 6-DoF nonlinear simulator tailored for a drone equipped with Magnus cylinders system. Through the execution of experimental flights, the simulator's performance is rigorously validated, establishing its reliability for future deployments.

keywords: Flight Dynamics Simulator, Flight Dynamics Modeling, Experimental Validations, Quadcopters, Magnus Effect.

1 INTRODUCTION

Recent years have seen the use of unmanned aerial vehicles (UAVs) is becoming increasingly prevalent across a wide range of industries and applications. One can cite industrial surveillance [1], infrastructure inspections [2], cinematography [3], merchandise transport [4] or aerial manipulation [5]. It is an ongoing challenge to develop UAVs that can fly longer distances and perform more complex tasks, and their power consumption is one of the main factors affecting their range and endurance. To address this issue, researchers have been exploring various methods for reducing UAVs energy consumption. In recent years, there has been renewed interest in the Magnus effect, which has been known for over a century [6], but has gained renewed attention in light of its potential application to UAVs [7, 8]. With Magnus cylinders attached to UAVs, lift can be generated without the use of traditional flight controls such as flaps or rudders, and flight trajectory can be controlled more precisely and flexibly.

Magnus cylinders offer several advantages over traditional control surfaces. These include that the speed and direction of rotation of the cylinders may be controlled robustly in order to create the desired aerodynamic forces, thereby providing greater control over the flight path of an aircraft. The Magnus effect can also reduce power consumption as it generates lift, which help maintain altitude, reducing the amount of power required to maintain the UAV's altitude. This is particularly relevant for UAVs that fly long distances over extended periods of time, such as those used in search

and rescue missions, etc. Overall, the Magnus effect has the potential to improve UAV technology by providing enhanced control and reducing energy consumption.

In this context, the presented study aims to develop a reliable 6-DoF nonlinear simulator for the quadcopter with the Magnus cylinder system. This will provide a comprehensive description of the system and help in the design of future autopilots in the applications of airborne wind energy production [9, 10].

Paper Organisation : Firstly, Section 2 addresses the flight mechanics model of the Magnus based quadcopter. It begins by presenting the equations of motion, covering both translational and rotational dynamics. In Section 3, a detailed explanation of all the forces and torques acting on the system is provided. The design of the PID based position control strategy is explained in Section 4. Moving on, Section 5 provides a brief description of the experimental setup used for validating the simulator. Finally, Section 6 presents an overview of the overall simulator along with the reliability study and simulator validation through analysing and comparing some simulations and experimental flight results.

Notations : for the seek of clarity, a series of notations is defined. For vector $\mathbf{x}$, we denote by $\|\mathbf{x}\|$ the $\mathcal{L}_2$ norm of $\mathbf{x}$ and by $\bar{\mathbf{x}}$ its transpose. $[\mathbf{x}]^{\mathcal{A}}$ is the representation of $\mathbf{x}$ expressed in $\mathcal{A}$ coordinate frame and $D^{\mathcal{A}}\mathbf{x}$ its derivative w.r.t frame $\mathcal{A}$. Regarding the dynamics, $\mathbf{s}_{BA}$ denotes the displacement vector of point B w.r.t point A , on the other hand, $\mathbf{S}_{BA}$ represents the skew matrix of the position vector $\mathbf{s}_{BA}$. Vector $\mathbf{v}_B^{\mathcal{A}}$ represents the linear velocity of point mass B w.r.t $\mathcal{A}$ coordinate frame and $\boldsymbol{\omega}^{\mathcal{B},\mathcal{A}}$ is the angular velocity vector of frame $\mathcal{B}$ w.r.t frame $\mathcal{A}$. Given an angle θ we denote c_θ , s_θ , and t_θ to the cosine, sin, and tangent of θ .

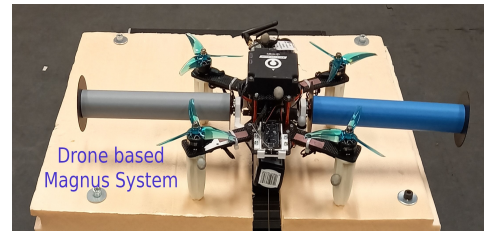


Figure 1: Magnus Effect-based Quadcopter prototype.

*Email address: zakeye.azaki@gipsa-lab.grenoble-inp.fr

2 NONLINEAR FLIGHT MECHANICS MODEL

The Magnus based quadcopter system consists of a quadrotor drone and two spinning Magnus cylinders connected to the right and left of the drone respectively as shown in Fig. 1. Table 1, represents the involved parameters of the system. The standard flight mechanics theory [11] is used to derive the 6-DoF nonlinear dynamics that govern the behavior of the system. All the following parameters and equations are used and coded in MATLAB/Simulink, to form the core of the realistic Magnus Effect-based Quadcopter System Simulator. The following reference frames are defined to formulate the equations of motion,

- Inertial Frame $\mathcal{I}(\mathbf{i}_1, \mathbf{i}_2, \mathbf{i}_3)$: Its base point I is assumed to be the reference of the position measurements.
- Drone Body Frame $\mathcal{D}'(\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3)$: Its base point coincides with the drone's center of mass D . The base vectors $\mathbf{d}_1$, $\mathbf{d}_2$, and $\mathbf{d}_3$ are aligned with the principle axes of the moment of inertia, such that $\mathbf{d}_3$ is directed upwards.
- Right Magnus Frame $\mathcal{M}_r(\mathbf{m}_{r1}, \mathbf{m}_{r2}, \mathbf{m}_{r3})$: Its base point coincides with the right Magnus wing center of mass M_r . The base vector $\mathbf{m}_{r2}$ is aligned with the right Magnus cylinder axis of rotations.
- Left Magnus Frame $\mathcal{M}_l(\mathbf{m}_{l1}, \mathbf{m}_{l2}, \mathbf{m}_{l3})$: Its base point coincides with the left Magnus wing center of mass M_l . The base vector $\mathbf{m}_{l2}$ is aligned with the left Magnus cylinder axis of rotations.
- System Body Frame $\mathcal{D}(\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3)$: Its base point coincides with the system drone+Magnus's center of mass C . The base vectors are parallel to that of $\mathcal{D}'$. This frame is considered as the body frame supporting all other spinning bodies.

The rotation matrix in this flight mechanics $\mathbf{R}^{\mathcal{D}\mathcal{I}}$ is the one of system frame $\mathcal{D}$ w.r.t inertial frame $\mathcal{I}$. It is composed of three rotations by the so-called Euler angles: roll, pitch, yaw or ϕ , θ and ψ . In our simulator, we use quaternions instead of Euler angles to represent the drone's rotation, as quaternions avoid the problem of gimbal lock and are more computationally efficient. The rotation quaternion is represented by the four dimensional coordinates $\bar{q} = [q_0 \ q_1 \ q_2 \ q_3]$.

Parameter	Description	Value
m	Total mass	1.568 kg
m_D	Drone mass	1.47 kg
m_{M_k}	kth Magnus mass	0.049 kg
L_{M_k}	kth Magnus length	0.179 m
R_{M_k}	kth Magnus radius	0.0175 m

Table 1: Model Parameters

It is assumed that the two unit directions $\mathbf{m}_{r2}$ and $\mathbf{m}_{l2}$ are collinear and they are parallel to $\mathbf{d}_2$ with an offset along $\mathbf{d}_3$ of δ_z such that,

$$[\overline{\mathbf{s}_{M_r D}}]^{\mathcal{D}} = [0 \quad \frac{L_r}{2} \quad -\delta_z] \quad , \quad [\overline{\mathbf{s}_{M_l D}}]^{\mathcal{D}} = [0 \quad -\frac{L_l}{2} \quad -\delta_z] \quad (2.1)$$

The common center of mass C can be determined as follows,

$$[\mathbf{s}_{CD}]^{\mathcal{D}} = \frac{\sum_k m_{M_k} [\mathbf{s}_{M_k D}]^{\mathcal{D}}}{m} \quad (2.2)$$

2.1 System Mathematical Model

First, it is important to mention that the centers of mass are mutually fixed. The translational and attitude dynamic equations are formulated using Newton's and Euler's laws of clustered bodies, respectively:

$$m \mathbf{D}^{\mathcal{I}} \mathbf{v}_C^{\mathcal{I}} = \mathbf{F}_C + \mathbf{F}_D + \sum_k \mathbf{F}_{M_k} + \mathbf{P} \quad (2.3)$$

$$\begin{aligned} \mathbf{D}^{\mathcal{I}}(\mathbf{I}_D^{\mathcal{D}} \mathbf{w}^{\mathcal{D}\mathcal{I}}) + \mathbf{D}^{\mathcal{I}}(m_D \overline{\mathbf{S}}_{DC} \mathbf{S}_{DC} \mathbf{w}^{\mathcal{D}\mathcal{I}}) + \sum_k \mathbf{D}^{\mathcal{I}}(\mathbf{I}_{M_k}^{\mathcal{M}_k} \mathbf{w}^{\mathcal{M}_k \mathcal{I}}) \\ + \sum_k \mathbf{D}^{\mathcal{I}}(m_{M_k} \overline{\mathbf{S}}_{M_k C} \mathbf{S}_{M_k C} \mathbf{w}^{\mathcal{M}_k \mathcal{I}}) = \mathbf{\Gamma}_C + \mathbf{\Gamma}_D + \sum_k \mathbf{\Gamma}_{M_k} \end{aligned} \quad (2.4)$$

where $k \in \{r, l\}$ represents the dynamics of the right and left Magnus separately and $\mathbf{P} = m\mathbf{g}$ is the system weight. The force vectors $\mathbf{F}_D$ and $\mathbf{F}_{M_k}$ are the total forces exerted on the system due to the drone propellers and Magnus cylinders respectively. Similarly, $\mathbf{\Gamma}_D$ and $\mathbf{\Gamma}_{M_k}$ are the torques acting on the system due to the drone propellers and Magnus cylinders respectively. However, the force vector $\mathbf{F}_C$ and the torque vector $\mathbf{\Gamma}_C$ are the additional control contribution. The detailed derivation of these forces and torques vectors will be represented in the next sections.

The angular velocity dynamics can be derived from (2.4) by

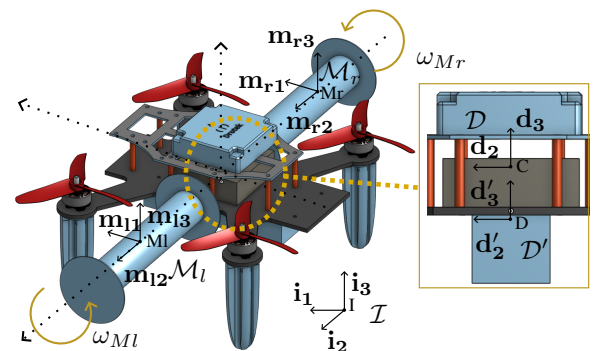


Figure 2: Frame definition on the quadcopter. From figure, $\mathcal{D}$ represents the body frame, $\mathcal{D}'$ the drone body frame, $\mathcal{M}_l$ the left Magnus frame, $\mathcal{M}_r$ the right Magnus frame and eventually $\mathcal{I}$ the inertial frame.

transferring the rotational derivative to the frame of the main body $\mathcal{D}$. We can deduce,

$$\begin{aligned} J D^{\mathcal{D}} \mathbf{w}^{\mathcal{D}\mathcal{I}} &= -\Omega^{\mathcal{D}\mathcal{I}} J \mathbf{w}^{\mathcal{D}\mathcal{I}} + \Gamma_C + \Gamma_D + \sum_k \Gamma_{M_k} \\ &- \sum_k (\Omega^{\mathcal{D}\mathcal{I}} \mathbf{I}_{M_k}^{\mathcal{M}_k} \mathbf{w}^{\mathcal{M}_k\mathcal{D}}) - \sum_k (\mathbf{I}_{M_k}^{\mathcal{M}_k} D^{\mathcal{D}} \mathbf{w}^{\mathcal{M}_k\mathcal{D}}) \end{aligned} \quad (2.5)$$

such that:

$$\begin{cases} \mathbf{J}_D^{\mathcal{D}} = \mathbf{I}_D^{\mathcal{D}'} + m_D \bar{\mathbf{S}}_{DC} \mathbf{S}_{DC} \\ \mathbf{J}_M^{\mathcal{D}} = \sum_k (\mathbf{I}_{M_k}^{\mathcal{M}_k} + m_{M_k} \bar{\mathbf{S}}_{M_k C} \mathbf{S}_{M_k C}), \\ \mathbf{J} = \mathbf{J}_D^{\mathcal{D}} + \mathbf{J}_M^{\mathcal{D}} \end{cases} \quad (2.6)$$

The revolving angular velocity vectors and the moment of inertia matrix of each Magnus cylinder about its axis of symmetry, are expressed as,

$$[\mathbf{w}^{\mathcal{M}_k\mathcal{D}}]^{\mathcal{D}} = \begin{bmatrix} 0 \\ w_{M_k} \\ 0 \end{bmatrix}, [\mathbf{I}_{M_k}^{\mathcal{M}_k}]^{\mathcal{D}} = \begin{bmatrix} I X_k & 0 & 0 \\ 0 & I Y_k & 0 \\ 0 & 0 & I X_k \end{bmatrix} \quad (2.7)$$

As a result, the translational and attitude dynamic state variables correspond to frame $\mathcal{D}$'s linear and angular velocities w.r.t. frame $\mathcal{I}$, respectively:

$$[\mathbf{v}_C^{\mathcal{I}}]^{\mathcal{I}} = [\dot{\mathbf{s}}_{CI}]^{\mathcal{I}} = \begin{bmatrix} \dot{x} \\ y \\ z \end{bmatrix}, [\mathbf{w}^{\mathcal{D}\mathcal{I}}]^{\mathcal{D}} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (2.8)$$

Moreover, the skew matrix of the angular velocity vector $\mathbf{w}^{\mathcal{D}\mathcal{I}}$ and the drone moment of inertia are expressed as:

$$\begin{aligned} [\Omega^{\mathcal{D}\mathcal{I}}]^{\mathcal{D}} &= \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}, \\ [\mathbf{I}_D^{\mathcal{D}'}]^{\mathcal{D}} &= \begin{bmatrix} I X & I X Y & I X Z \\ I X Y & I Y & I Y Z \\ I X Z & I Y Z & I Z \end{bmatrix} \end{aligned} \quad (2.9)$$

The equation of angular position can be expressed in terms of angular velocities $\mathbf{w}^{\mathcal{D}\mathcal{I}}$ expressed in $\mathcal{D}$. Based on the Euler ZYX formalism,

$$\begin{cases} \dot{\Theta} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \mathbf{W}^{-1} \mathbf{w}^{\mathcal{D}\mathcal{I}} \\ \mathbf{W}^{-1} = \begin{bmatrix} 1 & s_{\phi} t_{\theta} & c_{\phi} t_{\theta} \\ 0 & c_{\phi} & -s_{\phi} \\ 0 & s_{\phi}/c_{\theta} & c_{\phi}/c_{\theta} \end{bmatrix} \end{cases} \quad (2.10)$$

with $\mathbf{W}$ known as the Wronskien matrix of the Euler angles Θ attitude representation. The quaternion formalism of the angles' dynamics is obtained as follows,

$$\dot{\mathbf{q}} = \frac{1}{2} \begin{bmatrix} 0 & -\overline{\mathbf{w}^{\mathcal{D}\mathcal{I}}} \\ \mathbf{w}^{\mathcal{D}\mathcal{I}} & \Omega^{\mathcal{D}\mathcal{I}} \end{bmatrix} \mathbf{q} \quad (2.11)$$

3 FORCES AND TORQUES

3.1 Actuator dynamics

The presented system uses six brushless motors as actuators. Each one of which is modeled in this work by its single-phase electro-mechanical equivalent model, defined as follows for w the motor rotation speed :

$$\begin{cases} J_r \dot{w} &= \Gamma_{mot} - \Gamma_{res} \\ U &= \text{sat}(e + RI + LI) \end{cases} \quad (3.1)$$

Where J_r represents the inertia of the motor and load (Magnus or propeller) assembly. U , e and I are respectively referring to the motor phase voltage, the electromechanical force and the phase current. The phase resistance R and inductance L are directly measured on the motor. Motor torque Γ_{mot} is proportional to the phase current by electric constant K_c . Electromechanical force is proportional to the rotation speed through mechanical constant K_m . Resistive torque Γ_{res} can be approximated as the sum of an air friction quadratic torque, with C_Q the drag coefficient of the motor load, and a mechanical dry friction $\alpha\omega$:

$$\begin{cases} \Gamma_{mot} &= K_c I \\ \Gamma_{res} &= C_Q \omega^2 + \alpha\omega \\ e &= K_m \omega \end{cases} \quad (3.2)$$

In the rest of this paper, ω_{R_i} will refer to each propeller equipped motor $i \in \{1, 2, 3, 4\}$ rotation speed, and $\omega_{\mathcal{M}_k}$ to the Magnus cylinder equipped motor $k \in \{r, l\}$.

3.2 Drone Forces:

- The forces applied to the system by the four propellers $i \in \{1, 2, 3, 4\}$ are :

$$\mathbf{F}_{R_i}^m = C_T \|\mathbf{w}^{\mathcal{R}_i\mathcal{D}}\| \mathbf{w}^{\mathcal{R}_i\mathcal{D}} \quad (3.3)$$

with C_T generalized thrust coefficient and $\mathbf{w}^{\mathcal{R}_i\mathcal{D}}$ is the angular velocity vector of each rotor i such that,

$$\mathbf{w}^{\mathcal{R}_i\mathcal{D}} = w_{R_i} \mathbf{d}_3 \quad (3.4)$$

Therefore, the total thrust force exerted by the four propellers is

$$\mathbf{F}_D^m = \sum_i \mathbf{F}_{R_i}^m \quad (3.5)$$

- The aerodynamic forces due to its motion through the apparent wind speed, which is computed as

$$\mathbf{v}_a = \mathbf{v}_w - \mathbf{v}_C^{\mathcal{I}} \quad (3.6)$$

with $\mathbf{v}_w$ is the wind velocity vector.

We consider here only the aerodynamic drag forces of drone's body

$$\mathbf{F}_D^a = \frac{1}{2} \rho C_D \|\mathbf{v}_a\|^2 \mathbf{S} \quad (3.7)$$

with C_D is the drone's drag coefficient and $\mathbf{S}$ is the drone's exposed surface vector.

3.3 Magnus Forces:

The aerodynamic characteristics of the Magnus cylinder is affected by various factors. The most important one that controls Magnus effect-based wing is its spin ratio X which is the ratio between the local tangential velocity and the apparent wind velocity v_a , such that for each Magnus wing $k \in \{r, l\}$

$$X_k = \frac{R_{M_k} \|\mathbf{w}^{\mathcal{M}_k \mathcal{D}}\|}{\|\mathbf{v}_a\|} \quad (3.8)$$

In this work, we chose to add endplates to the two Magnus cylinders. This can significantly enhance lift and improve the lift-to-drag ratio while maintaining small aspect ratio $\Lambda = 5.1$. The endplates diameter was chosen to be twice that of the Magnus cylinder. The drag and lift coefficients dynamics are extracted from the wind tunnel tests gathered and analysed in [12]. These forces can be derived as follows:

$$\begin{aligned} \mathbf{F}_{M_k}^D &= \frac{1}{2} \rho C_{D_k} S_{M_k} \|\mathbf{v}_{a_k}\|^2 \mathbf{e}_{D_k} \\ \mathbf{F}_{M_k}^L &= \frac{1}{2} \rho C_{L_k} S_{M_k} \|\mathbf{v}_{a_k}\|^2 \mathbf{e}_{L_k} \end{aligned} \quad (3.9)$$

such that S_{M_k} represents the projected surface area of each $k \in \{r, l\}$ Magnus cylinder and the aerodynamic drag and lift coefficient of the right and left Magnus cylinders are as follows:

$$\begin{cases} C_{D_k} := C_{D_k}(\mathbf{w}^{\mathcal{M}_k \mathcal{D}}, \|\mathbf{v}_{a_k}\|) \\ C_{L_k} := C_{L_k}(\mathbf{w}^{\mathcal{M}_k \mathcal{D}}, \|\mathbf{v}_{a_k}\|) \end{cases} \quad (3.10)$$

The apparent wind velocity experienced by right and left cylinders at their respective center of mass M_r and M_l respectively are:

$$\begin{aligned} \mathbf{v}_{a_r} &= \mathbf{v}_a + \frac{L_{M_r}}{2} r \mathbf{d}_1 \\ \mathbf{v}_{a_l} &= \mathbf{v}_a - \frac{L_{M_l}}{2} r \mathbf{d}_1 \end{aligned} \quad (3.11)$$

The directions of the drag and lift forces for each Magnus wings $i \in \{r, l\}$ is defined such that the drag force is in the direction of the apparent wind velocity and the lift force is orthogonal to the Magnus wing axis of rotation and to the apparent wind velocity, then we deduce:

$$\begin{cases} \mathbf{e}_{D_k} = \frac{[\mathbf{v}_{a_k}]^{\mathcal{D}}}{\|\mathbf{v}_{a_k}\|} \\ \mathbf{e}_{L_k} = \mathbf{m}_{k2} \times \mathbf{e}_{D_k} \end{cases} \quad (3.12)$$

Hence, the total aerodynamic forces of each Magnus wing is read as:

$$\begin{cases} \mathbf{F}_{M_r}^a = \mathbf{F}_{M_r}^D + \mathbf{F}_{M_r}^L \\ \mathbf{F}_{M_l}^a = \mathbf{F}_{M_l}^D + \mathbf{F}_{M_l}^L \end{cases} \quad (3.13)$$

3.4 Total Forces:

We can deduce the total forces applied to the system, in inertial frame $\mathcal{I}$, based on (3.3)-(3.13) as follows:

$$\begin{aligned} \mathbf{F}_C &= \mathbf{R}^{\mathcal{DI}} \mathbf{F}_D^m \\ \mathbf{F}_D &= \mathbf{R}^{\mathcal{DI}} \mathbf{F}_D^a \\ \sum_k \mathbf{F}_{M_k} &= \mathbf{R}^{\mathcal{DI}} (\mathbf{F}_{M_r}^a + \mathbf{F}_{M_l}^a) \end{aligned} \quad (3.14)$$

3.5 Drone torques:

- Spinning drone torque:

The drone yaw torque is defined as follows,

$$\Gamma_D^y = \sum_i (C_Q \|\mathbf{w}^{\mathcal{R}_i \mathcal{D}}\| \mathbf{w}^{\mathcal{R}_i \mathcal{D}})$$

such that C_Q is the propeller's drag coefficient.

- Torque induced by drone's motors thrust forces is computed as follows:

$$\Gamma_D^m = \sum_i (\mathbf{s}_{R_i C} \times \mathbf{F}_{R_i}^m) \quad (3.15)$$

where $\mathbf{s}_{R_i C}$ for $i \in \{1, 2, 3, 4\}$ specifies the drone's geometry.

- Gyroscopic Effect drone torques:

As the drone's rotors $\mathcal{R}_i$ for $i \in \{1, 2, 3, 4\}$ is spinning around $\mathbf{d}_3$, then if the system is rolling or pitching, a gyroscopic torque is resulted as follows:

$$\Gamma_D^g = I_r \sum_i (\mathbf{w}^{\mathcal{R}_i \mathcal{D}} \times \mathbf{w}^{\mathcal{DI}}) \quad (3.16)$$

with I_r is the rotor and propeller moment of inertia.

- Inertial rotation torque:

$$\Gamma_D^i = -I_r \sum_i (D^{\mathcal{D}} \mathbf{w}^{\mathcal{R}_i \mathcal{D}}) \quad (3.17)$$

3.6 Magnus torques:

- Spinning torque:

The Magnus cylinder pitch torque is defined as follows,

$$\Gamma_M^p = \sum_k (C_{Q_k} \|\mathbf{w}^{\mathcal{M}_k \mathcal{D}}\| \mathbf{w}^{\mathcal{M}_k \mathcal{D}}) \quad (3.18)$$

such that $C_{Q_k} = \frac{1}{2} \rho \pi C_f S_{M_k} R_{M_k}^3$ and C_f is the skin friction coefficient of cylinder surface [13].

- Magnus cylinder Aerodynamic torques: torques arise due to the difference between the lift and drag produced by each cylinder and is expressed as:

$$\begin{cases} \Gamma_{M_r}^a = \mathbf{s}_{M_r C} \times \mathbf{F}_{M_r}^a \\ \Gamma_{M_l}^a = \mathbf{s}_{M_l C} \times \mathbf{F}_{M_l}^a \\ \Gamma_M^a = \Gamma_{M_r}^a + \Gamma_{M_l}^a \end{cases} \quad (3.19)$$

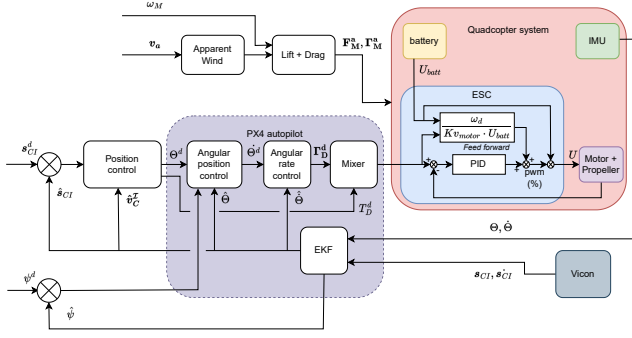


Figure 3: Magnus based quadcopter control architecture

- Gyroscopic Effect and inertial Magnus torques: We can see from (2.5) that the last two terms represent respectively the gyroscopic and inertial rotation torques of the two Magnus cylinders such that

$$\begin{aligned}\Gamma_M^g &= -\sum_k (\Omega^{DI} I_{M_k}^{\mathcal{M}_k} \omega^{\mathcal{M}_k \mathcal{D}}) \\ \Gamma_M^i &= -\sum_k (I_{M_k}^{\mathcal{M}_k} D^{\mathcal{D}} \omega^{\mathcal{M}_k \mathcal{D}})\end{aligned}\quad (3.20)$$

3.7 Total torques:

Torques from (3.20) could be included in torques exerted by Magnus cylinders and thus the angular velocity dynamics (2.5) can be simplified,

$$JD^{\mathcal{D}} \omega^{\mathcal{D}\mathcal{I}} = -\Omega^{\mathcal{D}\mathcal{I}} J \omega^{\mathcal{D}\mathcal{I}} + \Gamma_C + \Gamma_D + \sum_k \Gamma_{M_k} \quad (3.21)$$

Therefore, we can deduce the total torques applied to the system, in body frame $\mathcal{D}$, based on (3.16)-(3.15) as follows:

$$\begin{aligned}\Gamma_C &= \Gamma_D^y + \Gamma_D^m \\ \Gamma_D &= \Gamma_D^g + \Gamma_D^i \\ \sum_k \Gamma_{M_k} &= \Gamma_M^p + \Gamma_M^a + \Gamma_M^g + \Gamma_M^i\end{aligned}\quad (3.22)$$

4 CONTROL STRATEGY

In this section we present the design of the overall control strategy. The control strategy is based on simplified model of the system. This simplified model comes from a simplification of the complete nonlinear model, described in sections (2.3) and (3.21), in which aerodynamic effects, ground effect and gyroscopic effects are neglected. This model is described as follows:

$$\begin{cases} \dot{s}_{CI} &= v_C^{\mathcal{I}} \\ m \dot{v}_C^{\mathcal{I}} &= F_C + P \\ \dot{\Theta} &= W^{-1} \omega^{\mathcal{D}\mathcal{I}} \\ J \dot{\omega}^{\mathcal{D}\mathcal{I}} &= -\Omega^{\mathcal{D}\mathcal{I}} J \omega^{\mathcal{D}\mathcal{I}} + \Gamma_C \end{cases} \quad (4.1)$$

4.1 Position and Velocity Loop:

The system's position and velocity can be controlled by F_C . The latest is represented by its projections in the inertial frame $\mathcal{I}$: $F_{C_x}, F_{C_y}, F_{C_z}$ that controls x, y, z loops respectively. We have implemented the PID control strategy to compute these control forces, for each loop $q \in \{x, y, z\}$, as follows:

$$F_{C_q} = m(k_{d_q} \dot{q} + k_{p_q} q + k_{i_q} \int e_q d\tau) \quad (4.2)$$

with e_q is the tracking error of each loop $q \in \{x, y, z\}$. The inner loops are fed by the desired thrust force T_D^d and the desired Euler angles Θ^d . These are derived from (4.2) according to the kinematic transformation as follows:

$$\begin{cases} T_D^d &= m(F_{C_x} + g)/(c_\phi c_\theta) \\ \theta^d &= \text{atan2}((F_{C_x} c_\psi + F_{C_y} s_\psi), F_{C_z} + g), \\ \phi^d &= \text{atan2}(c_\theta((F_{C_x} s_\psi - F_{C_y} c_\psi), F_{C_z} + g)\end{cases} \quad (4.3)$$

4.2 Attitude and Angular Velocities Loops:

The simplified model in (4.1), gives a general view about the inner attitude and angular velocities control loops. On one hand, the angular position represented by the Euler angles Θ can be controlled by the angular velocities $\omega^{\mathcal{D}\mathcal{I}}$. On the other hand, the angular velocities $\omega^{\mathcal{D}\mathcal{I}}$ can be controlled by the controller torque Γ_C . We use a PX4 onboard autopilot [14]. This autopilot manages the attitude and angular speed loops. PX4 control structure for rate and attitude loops have been copied in the MATLAB/Simulink simulator, based on in-flight tune control gains and control diagrams given by PX4.

4.3 PX4 Mixers

- PX4 Normalized Mixer: The hover compensation is applied to account for any variations in the drone's hover performance. The normalized desired total thrust force is computed as, $T_D^n = \frac{hc}{mg} T_D$ with hc is the hover compensation factor dependent on the specific drone and its configuration.
- PX4 Identified Mixer: It is based on a linear relationship between torque Γ_C and force T_D commands and the four rotors PWM signals setpoint. It is described as:

$$\begin{bmatrix} \omega_{R_1} \\ \omega_{R_2} \\ \omega_{R_3} \\ \omega_{R_4} \end{bmatrix} = M_{PX4} \begin{bmatrix} \Gamma_C \\ T_D^n \end{bmatrix} \quad (4.4)$$

5 EXPERIMENTAL SETUP

Our custom build quadcopter flies on an Hollybro Pixhawk 4 flight controller running PX4 Autopilot. Offboard position control is performed on a ground station through ROS1. Communication between the UAV and the ground station is performed via Mavlink protocol through Wifi. In order to perform indoor flight as GPS (GNSS receiver) is not available, position measurement is provided by Vicon motion capture system. The UAV is running on a 4S LiPo battery. All material is listed in Table 2 and diagrammed in Figure 4

Brushless motor speed control is performed with an ESC (Electronic Speed Controller). The ESC is meant to apply to the motor a fraction of the battery voltage. This fraction is given by a standard digital input (most generally PWM or Dshot signal). Thus, standard motor control is performed in *open loop* with respect to the desired rotation speed. In practice, air and dry friction cause the motor to run slower than expected. In order to achieve precise speed control, and thus precise force control on each motor, we have implemented a custom firmware into standard ARM32 processor based ESCs allowing *close loop* speed control. In our experimental setup, the input of the ESC is then a desired speed, and not a percentage of battery voltage. In the ESC, speed control is performed through a standard PID regulation in which speed measurement is performed by monitoring the inversion of phase current due to the movement of rotor magnets. The ESC firmware that has been used is available at ¹.

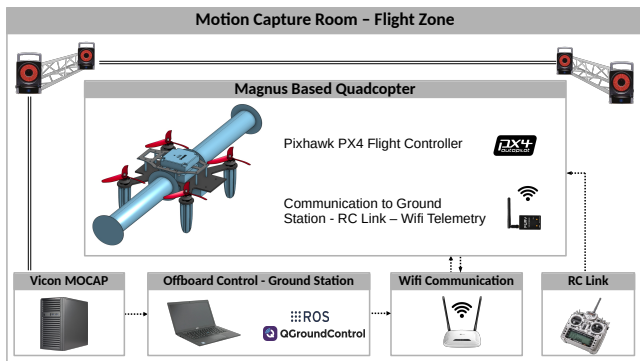


Figure 4: Gipsa-Lab experimental setup

6 RELIABILITY STUDY

To study the reliability of our model we propose a three steps approach to compare flight data to simulated ones. Propulsion model is to be validated with a dedicated protocol detailed in Section 6.1. On a position step scenario with no spinning of the cylinders, we will validate inner loops, inertia and body dynamics. Finally, added gyroscopic effects

¹<https://github.com/gipsa-lab-uav/AM32-MultiRotor-ESC-firmware>

Item	Description
Flight controller	Holybro Pixhawk 4 with PX4 v1.14 firmware
Battery	Bashing 4S 5000mAh
Radiocontroller	FrSky Taranis X9D
RC receiver	FrSky XM+
Motor	T-Motor F60proV 1750 Kv
Propeller	T-Motor T5147
Propeller ESC	HGLRC 4in1 Zeus 45A
Magnus ESC	HGLRC T-Rex 35A
Wifi Communication	ESP32 dev kit with serial/wifi bridge firmware
Motion capture system	12 Vicon T40s cameras, Tracker software

Table 2: Hardware setup

will be assessed under constant cylinder rotation speed scenario. Inertial effects will be shown in the inner attitude and rate loops by changing in flight the rotational speed of the cylinders. Modeling the aerodynamics of the Magnus cylinders' necessitates significant linear and rotational speed testing scenarios which are not performed in this work. In lift and drag aerodynamics are not addressed in this work as they are well documented in literature. We are then focusing on low linear speed scenarios. These torques and forces need significant linear speed and rotational speed. At this stage of the study, we based on the Magnus aerodynamic model already published as stated in Section 3.3. However, as future perspective, this dynamics will be validated during external flights at higher speeds.

6.1 Propulsion model validation

The propulsion model, defined in Section 3.1 as a model of a single-phase motor combined to a 5 inches propeller, has been validated experimentally. As explained in Section 4.3, the PX4 autopilot performs a linear mix between the desired forces from position controller and the desired motor speed fed to the ESC. However, as stated in Section 3.1, the generated force is not linear but proportional to the squared rotation velocity of the propeller.

In our control law, we work with a linear approximation of the thrust around the equilibrium point of hover flight for a given flight mass. In order to experimentally validate propeller model used in (3.3), we maintained the UAV in a hover flight with a position control based on the linear approximation of the thrust. Then, every 10 seconds an additional mass of 60 grams is added to the UAV. We monitored desired motor speeds [rad.s⁻¹] and desired Thrust [N]. Figure 5 shows the experimental validation of the propulsion model and of its linear approximation around the flight mass equilibrium point : the estimation of the thrust as proportional to the sum of squared velocities of propellers matches the actual mass of the UAV, whereas its linear approximation used for control fits with the real value around the real flight mass. The lift

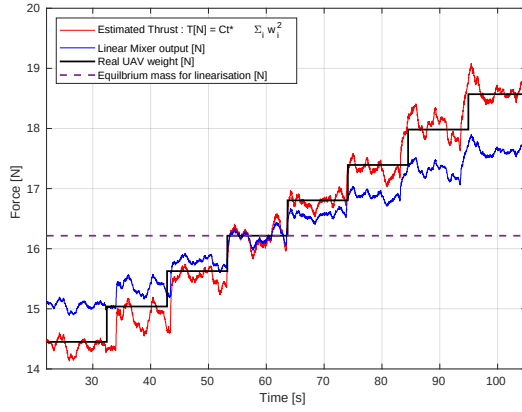


Figure 5: Experimental Thrust model validation

coefficient C_T has been calculated from desired motor speeds and real UAV weight during the protocol.

6.2 Dynamical Model without Magnus rotation

Figures 6, 7, 8 and 9 show comparison between flight data and simulator data of all nested loops dynamics over the same 3D position steps scenario and no cylinder rotation. A yaw step is performed at $t = 110s$. Even if noise level over the angular rates and attitude loops, Figure 6 and 7, are underestimated in the simulator, the main dynamics remains correctly predicted. These noises are mainly due to unmodelled vibrations on the UAV frame, and unmodelled aerodynamics disturbances in an indoor environment (wind turbulence).

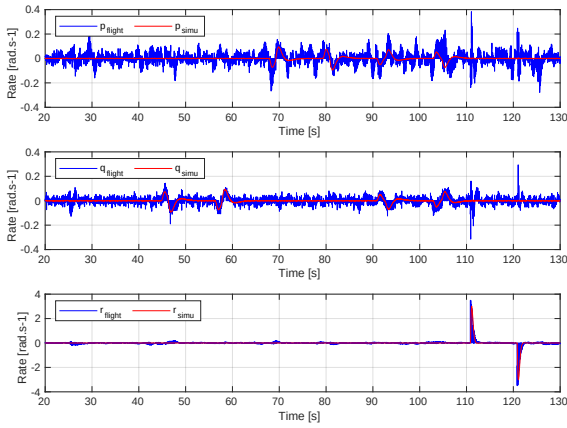


Figure 6: Angular velocities response in case of no Magnus rotation

6.3 Magnus cylinder induced disturbances

- To highlight the gyroscopic effects due to cylinders rotation, let's focus on the effect of a yaw step in hovering flight, at constant cylinder speed $\omega_{M_r} = \omega_{M_l} = 7200\text{rpm}$ over the angular rate p , as shown in Figure 10.

- To highlight the effects of inertia due to cylinder rotation, let's vary cylinder speed in hovering flight between

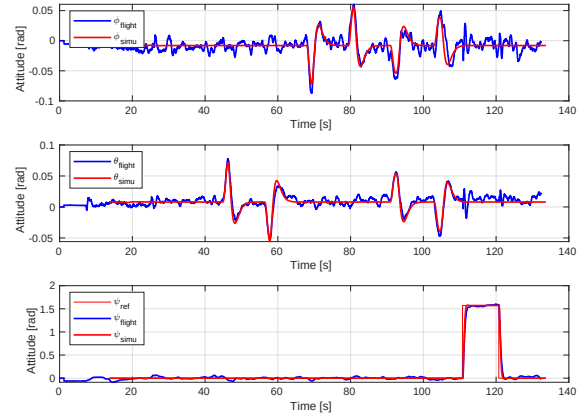


Figure 7: Euler angles response in case of no Magnus rotation

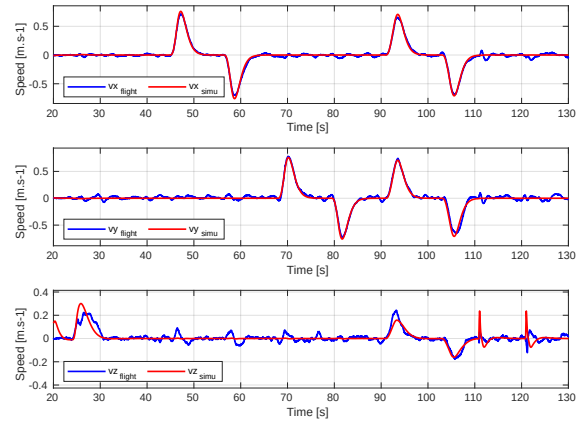


Figure 8: Linear velocities response in case of no Magnus rotation

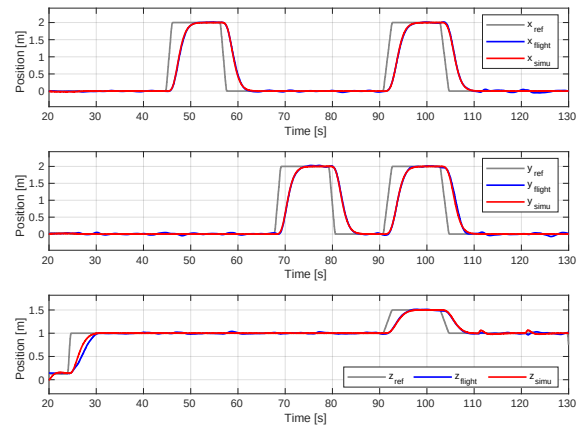
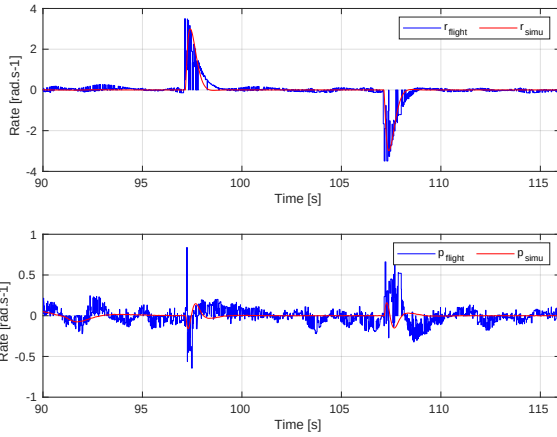
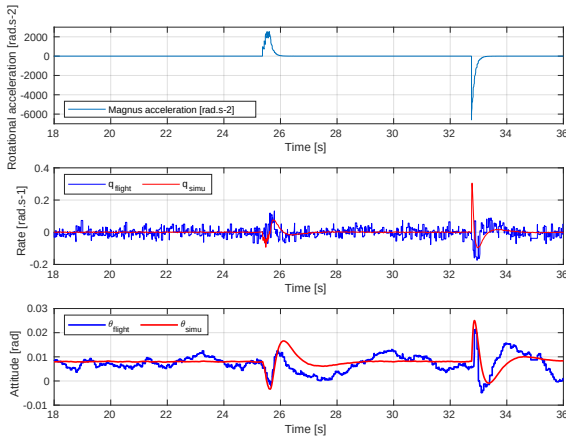


Figure 9: Positions response in case of no Magnus rotation

$\omega_{M_r} = \omega_{M_l} = 5100\text{ rpm}$ (rotations per minutes) and $\omega_{M_r} = \omega_{M_l} = 11400\text{ rpm}$. We simulate and monitor an added inertial torque among the body d_2 axis as shown in Figure 11, on the rate and attitude loops. The acceleration of the cylinders is shown in top of the figure.

Figure 10: Magnus gyroscopic effect on p rateFigure 11: Magnus inertial effects among body d_2 axis

The Magnus rotor friction torque is estimated to be $1.8e - 3$ Nm for maximal speed of rotation 11400 rpm which is negligible regarding to other torques acting on the system.

7 CONCLUSION

This paper presented the design and experimental validation of a 6-DoF simulator for a Magnus-based quadcopter system. This simulator was validated based on flight experimental tests in an indoor environment and loop by loop validation. The results demonstrate that the simulator accurately captures the main dynamics of the system. Its reliability and compatibility with experimental data make it a tool for researchers and engineers to optimize the design and control strategies of Magnus-based quadcopter systems. The findings contribute in the field of Magnus-based quadcopters and enable their use in various industries, paving the way for efficient and capable unmanned aerial vehicles. For future work, it is recommended to incorporate more sensors and Extended Kalman Filter and validate the simulator based on outdoor experiments with higher speeds maneuvers.

ACKNOWLEDGEMENTS

This work has been partially supported by ROBOTEX 2.0 (Grants ROBOTEX ANR-10-EQPX-44-01 and TIRREX ANR-21-ESRE-0015) funded by the French program Investissements d'avenir.

REFERENCES

- [1] Giuseppe Silano, Jan Bednar, Tiago Nascimento, Jesus Capitan, Martin Saska, and Anibal Ollero. A multi-layer software architecture for aerial cognitive multi-robot systems in power line inspection tasks. In *ICUAS*, 2021.
- [2] Weibin Gu, Dewen Hu, Liang Cheng, Yabing Cao, Alessandro Rizzo, and Kimon P. Valavanis. Autonomous wind turbine inspection using a quadrotor. In *ICUAS*, 2020.
- [3] Arturo Torres-González, Jesús Capitán, Rita Cunha, Anibal Ollero, and Ioannis Mademlis. A multidrone approach for autonomous cinematography planning. In *ROBOT 2017: Third Iberian Robotics Conference*.
- [4] David Schneider. The delivery drones are coming. *IEEE Spectrum*, 57(1):28–29, January 2020.
- [5] Suseong Kim, Seungwon Choi, and H. Jin Kim. Aerial manipulation using a quadrotor with a two DOF robotic arm. In *2013 IEEE/RSJ*.
- [6] Jost Seifert. A review of the magnus effect in aeronautics. *Progress in Aerospace Sciences*, 55:17–45, 2012.
- [7] Yashank Gupta, Jonathan Dumon, and Ahmad Hably. Modeling and control of a magnus effect-based airborne wind energy system in cross-wind maneuvers. *IFAC-PapersOnLine*, 50(1):13878–13885, 2017.
- [8] Zakeye Azaki, Jonathan Dumon, Nacim Meslem, Ahmad Hably, and Pierre Susbielle. Modelling and control of a tethered drone for an awe application. In *ICCAD 2022*, 2022.
- [9] Lorenzo Fagiano, Manfred Quack, Florian Bauer, Lode Cernel, and Espen Oland. Autonomous airborne wind energy systems: accomplishments and challenges. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:603–631, 2022.
- [10] Zakeye Azaki, Jonathan Dumon, Nacim Meslem, and Ahmad Hably. Sliding mode control of tethered drone: Take-off and landing under turbulent wind conditions. In *ICUAS*, 2023.
- [11] P. H. Zipfel. *Modeling and Simulation of Aerospace Vehicle Dynamics*. American Institute of Aeronautics and Astronautics, Inc., 3rd edition, 2014.
- [12] C. Badalamenti, , and S. Prince. Effects of endplates on a rotating cylinder in crossflow. *Applied Aerodynamin Conference*, 2008.
- [13] C. Badalamenti. On the application of rotating cylinders to micro air vehicle. *Unpublished Doctoral thesis, City University London*, 2010.
- [14] Dario Brescianini, Markus Hehn, and Raffaello D'Andrea. Nonlinear quadcopter attitude control: Technical report. Technical report, 2013.

Numerical Investigation on Flow Control of Leading-Edge Reinforced Flexible Membrane Wing

Z. Liu^{1*} and WH. Xie²

State Key Laboratory of Mechanical Structural Strength and Vibration, Xi'an Jiaotong University, Xi'an, Shaanxi, China

ABSTRACT

The leading-edge reinforced flexible membrane wing can reduce spanwise deformation and has the advantage of adaptive passive deformation of the membrane wing. A membrane wing with a certain thickness at the leading edge and an equal thickness at the trailing edge was designed. The membrane wing film thickness is 1mm and the ratio of the film area to the whole wing is 72.2%. The Fluid-Structure Interaction (FSI) analysis method was established, and the aerodynamic characteristics of the wing was studied. Results showed that the maximum deformation of the leading edge of the wing tip is 4.75%. The flow separation is delayed due to the deformation of the elastic film which improves the aerodynamic performance after stall. It is found that the leading-edge reinforced membrane wing deformed little in spanwise direction which reduces the deterioration of aerodynamic performance caused by spanwise deformation. The membrane wing near the trailing edge still has significant adaptive passive deformation resulting in the improvement of stall performance.

1 INTRODUCTION

In recent years, there are more and more researches on deformable wings, whose main goal is to improve the aerodynamic characteristics of traditional rigid wings, improve efficiency and expand the flight envelope. The traditional wing deformation concept mainly focuses on the deformation of flaps, ailerons and wing sweepback. Although the efficiency and flight envelope of the wing have been greatly improved, the added mass has increased, and the aerodynamic performance has declined due to discontinuities and gaps between components, which is particularly obvious in low-speed flight. The lightweight, low cost, and simple structure of membrane wings make them more promising in low-speed applications compared to traditional rigid wings [1]. In the mid-20th century, the Princeton sail appeared, also known as the double membrane sail, which was basically composed of a rigid leading edge spar and a trailing edge wire (or spar). The membrane wraps around the leading and trailing edges,

forming the upper and lower wing surfaces. Originally conceived as an advanced sail, it was later transformed into an aircraft wing. Reference [2] conducted experimental tests on different sail configurations, and the results showed that the aerodynamic characteristics of membrane wings were superior to traditional rigid wings in terms of maximum lift and maximum lift drag ratio. A notable feature of membrane wings is their ability to naturally adjust their shape to adapt to constantly changing flow conditions, namely 'adaptive' passive deformation, resulting in excellent stall characteristics. In the study of membrane wings, many scholars have focused on the influence of the deformation of elastic membranes on aerodynamic performance. In their research, flat membrane wings with fixed peripheral frames are often used as research objects. For example, reference [3] studied the influence of membrane permeability characteristics on wing aerodynamic efficiency, and the results verified the possibility of dynamic pressure compensation through wing permeability. Fujita [4] proposed the "self consistent theory" and used it to predict the aerodynamic performance of flat membrane wings with fixed and free wingtips. This theory can effectively predict the deformation of elastic membranes and the lift hysteresis characteristics of flat membrane wings, and has good consistency with wind tunnel test data. Song [5] proposed a theoretical model for membrane bending caused by aerodynamic loads, indicating that the appropriate dimensionless parameter to describe the problem is the Weber number, which compares aerodynamic loads with membrane elasticity. It was found that the theory was very consistent with the experiment. Aerodynamic performance measurement shows that compared with rigid wings, flexible wings have higher lift slope, maximum lift coefficient and delayed stall at higher angles of attack. In addition, they exhibit strong hysteresis near zero angle of attack and stall angle. Mueller [6, 7] conducted research and analysis on curved flat membrane wings, demonstrating the importance of curvature and wing shape, indicating that the inverse Zimmerman shape is superior for low Re flow. Albertani [8] applied membrane wings to micro fixed wing unmanned aerial vehicles, developed and manufactured low aspect ratio wings, using an elastic latex film skin covered with a thin carbon fiber skeleton. The wings were tested in a low-speed wind tunnel facility that integrates a visual image correlation (VIC) system and a six component strain gauge balance. The static response characteristics of three different wing designs were measured, including full field displacement and plane strain measurements. The full field deforma-

*Email address(es): liuz@mail.xjtu.edu.cn

tion results indicate how the flexibility of the passive wing better affects aerodynamic performance compared to rigid models with similar geometric shapes. Measure elastic deformation and aerodynamic coefficients through wind tunnel testing methods. Previous studies have shown that membrane wings have good aerodynamic performance at low Re , but when aerodynamic loads are high, membrane wings exhibit significant spanwise bending and torsion deformation, which is detrimental to aerodynamic performance. The main reason is limited by the small thickness of the membrane wing, and excessive size of the reinforcing ribs can easily cause discontinuity between the reinforcing ribs and the membrane wing. Therefore, the small thickness of the reinforcing ribs is prone to spanwise bending and torsion deformation. In response to this issue, this article proposes a hybrid membrane wing with leading edge reinforcement, which combines traditional airfoils and membrane wings to enhance spanwise stiffness while retaining the advantages of membrane wings' adaptive passive deformation. This article conducts a study on structural deformation based on the thickness and membrane ratio of this airfoil.

Due to the small stiffness of membrane wings, they undergo significant "adaptive" passive deformation under aerodynamic loads. The large deformation of the structure has a significant impact on the flow field, which in turn affects the aerodynamic performance of the wing. This is a typical fluid structure coupling problem. At present, there are two main methods for solving fluid solid coupling problems: direct coupled solution and partitioned solution. The direct coupling method solves the fluid solid control equations by coupling them into the same equation matrix, and simultaneously solving the control equations of the fluid and solid in the same solver. However, in practical applications, direct solutions face convergence difficulties and time-consuming issues in synchronous solutions. Therefore, they are currently mainly used in simple problems such as electromagnetic structural coupling and thermal structural coupling, such as piezoelectric material simulation. The separation method does not require coupling the fluid solid control equations, but solves the fluid control equations and the solid control equations separately in the same solver or different solvers in the set order. The calculation results of the fluid domain and the solid domain are exchanged and transmitted through the fluid solid interface (FS Interface). When the convergence at this moment meets the requirements, proceed with the calculation at the next moment and obtain the final result sequentially. Compared with the direct coupling method, the separation method has the disadvantages of time lag and incomplete conservation of energy on the coupling interface. However, the advantages of this method are also obvious. It can maximize the use of existing methods and programs of computational fluid dynamics and computational solid mechanics, and only need to make a few modifications to them, so as to maintain the modularization of the program.

The research results indicate that at high reduction frequencies, flexible airfoils have the characteristic of low drag, and their aerodynamic performance is significantly superior to that of rigid airfoils; The passive deformation of flexible airfoils can suppress the generation of surface vortices and optimize aerodynamic characteristics. Some researchers conducted wind tunnel experiments on flexible wings with high aspect ratio, studying their physical parameters such as lift resistance, displacement, vortex shedding frequency, suction surface, and wake vorticity field. Through experimental research, it was found that flexible wings with high aspect ratio exhibit similar characteristics to rigid wings under the same conditions in some aspects. In this study, due to the small stiffness of the elastic thin film and its significant deformation under aerodynamic loads, it is necessary to use the fluid solid coupling method for research. Considering the computational cost and convergence, a weak coupling solution method, namely the separation method, is adopted.

This article proposes a design concept for a leading edge reinforced bioinspired membrane wing, and establishes an aerodynamic model for the leading edge reinforced bioinspired membrane wing based on a structured grid flow field partitioning method. The finite element model for the leading edge reinforced bioinspired passive membrane wing is also developed, and flow control research on the leading edge reinforced bioinspired passive membrane wing is carried out through fluid structure coupling method.

2 COMPUTATIONAL METHODOLOGY

Establish a bidirectional FSI model for structural aerodynamics for further analysis. The structure aerodynamic bidirectional FSI model mainly includes structural model, aerodynamic flow field model, and structure aerodynamic data interaction platform. The structural model is a finite element model, which calculates the displacement deformation of the wing under aerodynamic loads and transmits the displacement deformation to the aerodynamic model through the structural aerodynamic data interaction platform; The aerodynamic model recalculates the flow field based on displacement deformation, obtains aerodynamic data (mainly aerodynamic pressure), and transmits it to the structure through the structure aerodynamic interaction platform; The structure recalculates displacement and deformation, iterating continuously until the residual meets the accuracy requirements. The flight speed in this article is between 15m/s and 20m/s, therefore it is considered that the surrounding fluid is incompressible. At the same time, its operating state is at low Reynolds number, and viscous force has a greater impact, so viscous effect is considered. A three-dimensional incompressible Navier Stokes control equation considering viscous effects for a hybrid membrane wing with reinforced leading edge is obtained comprehensively. The $k-\omega$ SST turbulence model is used. And a second-order discretization scheme was used as well. For membrane wing structure, the finite element

method is used to discretize the structure in space, and Newmark method is used to solve it. Ansys software was used for the simulation.

3 COMPUTATIONAL MODEL OF LEADING-EDGE REINFORCED FLEXIBLE MEMBRANE WING

3.1 Introduction of Leading-Edge Reinforced Flexible Membrane Wing

When the flight Reynolds number is low, the membrane wing has better aerodynamic performance than the traditional airfoil. Traditional membrane wings are all curved thin plate structures with equal thickness. To enhance their load-bearing capacity, reinforcing ribs are added to the membrane wings. Generally, cylindrical reinforcing ribs are used at the leading edge, while narrow strip thin plate reinforcing ribs are used for the rest. When the aerodynamic load is small, this layout can meet the required bearing capacity, but when the aerodynamic load is large, the membrane wing will produce significant bending and torsional deformation, which reduces the aerodynamic performance of the membrane wing. Therefore, this article combines traditional airfoils with membrane wings, retaining the thickness of the leading edge of the traditional airfoil. The half of the airfoil adopts membrane wing design, resulting in a hybrid airfoil with strengthened leading edge, as shown in Figure 1.

Based on the above airfoil, a leading edge reinforced membrane wing was designed as shown in Figure 2. The wing has a span length of 0.6mm and a chord length of 0.12mm. The yellow part in the figure shows the surrounding support structure, composed of carbon fiber reinforced composite materials, which is the main structure of the leading edge reinforced hybrid membrane wing. The blue part in the figure is an elastic film composed of Dielectric elastomers (DE) material, mainly responsible for the adaptive passive deformation function of the membrane wing. The properties of DE materials and carbon fiber reinforced composite materials are shown in Table 1 and Table 2.

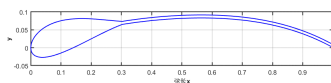


Figure 1: Leading-edge reinforced airfoil.

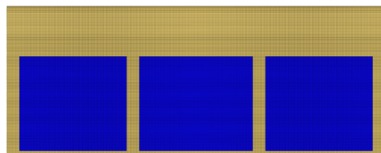


Figure 2: Leading-edge reinforced membrane wing.

$\rho(kg/m^3)$	$E_1(MPa)$	μ
960	0.748	0.47

Table 1: Material parameters of DE material.

$\rho(kg/m^3)$	$E_x(Pa)$	$E_y(Pa)$	$E_z(Pa)$	μ_{xy}
1300	4.2e10	4.2e10	1e10	0.35
μ_{yz}	μ_{xz}	$G_{zy}(Pa)$	$G_{yz}(Pa)$	$G_{xz}(Pa)$
0.3	0.3	2.1e10	7e9	7e9

Table 2: Material parameters of carbon fiber reinforced composites.

3.2 Aerodynamic Model and Structural Model

A three-dimensional aerodynamic model of the leading edge reinforced membrane wing was established, as shown in Figure 3. The front flow field of the wing is 10c, the rear flow field is 20c, and the upper and lower flow fields are 10c. The total number of grids is 7557116, all of which are hexahedral grids. The number of boundary layers is 10, and the thickness of the first layer is $y^+=1$, i.e. 0.019mm. Establish a finite element model of the wing structure based on the geometry of the membrane wing, using hexahedral elements as the mesh type, which can truly reflect structural deformation and establish numerical interpolation relationships with the flow field, as shown in Figure 4. The middle part is an elastic film, and the surrounding part is the skeleton support structure of the membrane wing. The elastic film and the skeleton support structure are connected through the side, and the Bounded connection relationship is given to ensure consistent calculation efficiency and accuracy of the connection nodes. The strengthening part of the leading edge of the membrane wing is the connection part between the wing and the fuselage, so fixed constraints are given on the side of the wing root at the strengthening part of the leading edge.

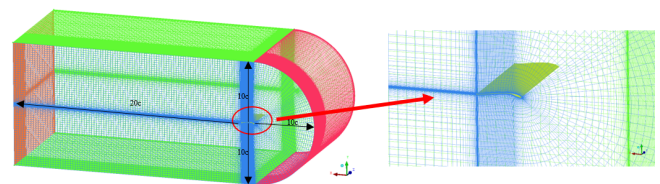


Figure 3: Aerodynamic mesh of the wing.

4 RESULTS AND DISCUSSION

In order to understand the aerodynamic characteristics of the membrane wing, the aerodynamic characteristics of rigid wing which did not consider the elasticity of wing and membrane wing were studied separately. The Reynolds number is 122262. Because the stall angle of the wing is about 12°, the range of the angle of attack is from 10° to 20°. Figures 5,

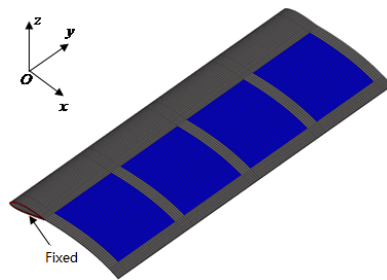


Figure 4: Structural mesh of the wing.

6 and 7 shows the lift coefficient curve, drag coefficient curve and lift drag ratio curve of rigid wing and membrane wing at different angles of attacks.

Compared with the rigid wing aerodynamic results, the lift coefficient of the rigid wing before stall is greater than the lift coefficient of the leading edge strengthened hybrid membrane wing, which is mainly due to the passive deformation of the membrane wing under the effect of aerodynamic pressure. When it is greater than the stall angle of attack, the lift coefficient of the rigid wing is less than that of the leading edge strengthened hybrid film wing, and the stall angle of attack of the rigid wing is around 12° , the stall angle of attack of the leading edge strengthened hybrid film wing is around 14° , and the stall angle of attack is delayed by about 2° . By comparing the lift coefficient decline after stall, it can be seen that the lift coefficient of the leading edge strengthened hybrid film wing decreases slightly, while the lift coefficient of the rigid wing decreases greatly. By comparing the drag coefficient, it can be seen that when the angle of attack is small, the drag coefficient of the leading edge reinforced mixed film wing is smaller than that of the rigid wing. By comparing the lift to drag ratio, it can be seen that the lift to drag ratio of the leading edge reinforced mixed film wing is greater than that of the rigid wing throughout the entire process, resulting in higher aerodynamic efficiency. Verified that the hybrid membrane wing with reinforced leading edge still retains the advantages of adaptive passive deformation of the membrane wing.

Figure 8 shows the pressure contour at the middle section of a rigid wing and a leading edge reinforced mixed membrane wing at different angles of attack, where a is the rigid wing and b is the leading edge reinforced mixed membrane wing. According to the pressure cloud map, the pressure difference between the upper and lower surfaces of the wing is mainly concentrated at the leading edge, so the lift is mainly distributed at the leading edge. However, due to the traditional airfoil used for strengthening the leading edge, it has strong resistance to spanwise bending deformation and has small displacement deformation at the leading edge of the wing tip. Comparing the pressure cloud maps of a rigid wing and a hybrid membrane wing with a strengthened lead-

ing edge after stall (angle of attack greater than 14°), it can be observed that the deformation of the membrane wing results in a smaller range of low-pressure areas on the upper surface of the wing and a greater lift towards the leading edge.

Figure 9 shows the Q-criterion vorticity contour of a rigid wing and a leading edge reinforced mixed membrane wing at different angles of attack, where a is the Q-criterion vorticity contour of the middle section position of the rigid wing, b is the Q-criterion vorticity contour of the middle section position of the leading edge reinforced mixed membrane wing, and c is the three-dimensional Q-criterion vorticity contour of the leading edge reinforced mixed membrane wing. Compared with the vorticity nephogram at the middle section of the rigid wing and the leading edge strengthened hybrid film wing under $AOA=14^\circ$, it can be seen that the vortex area on the upper surface of the rigid wing is large, and flow separation occurs on the upper surface, so the lift coefficient decreases; However, due to the deformation of the elastic membrane, the vortex area of the leading edge strengthened hybrid membrane wing is reduced, and the air flow is still attached to the upper surface of the wing. Therefore, the flow separation is not serious, and the lift coefficient remains large, delaying the occurrence of flow separation. Comparing the vorticity contour after stall, it can be seen that the vortex area of the leading edge reinforced mixed membrane wing is smaller than that of the rigid wing, and the deformation of the elastic membrane improves the aerodynamic performance after stall. According to the three-dimensional Q-criterion vorticity contour of the hybrid membrane wing with strengthened leading edge, flow separation first occurs at the wing root and gradually spreads to the wing tip as the angle of attack increases; The vortex region is concentrated at the position of the elastic film. For example, when $AOA=14^\circ$, except for the wing tip vortex, the vortex is mainly concentrated at the position of the elastic film near the wing root. When $AOA=16^\circ$, the vortex is mainly concentrated at the position of the elastic film near the wing root and in the middle, but the vortex near the wing root is larger. When $AOA=18^\circ$, the vortex is mainly concentrated at three positions of the elastic film, and gradually decreases from the wing root to the wing tip.

To evaluate the ability of the hybrid membrane wing with reinforced leading edge to resist spanwise bending and torsion deformation, it is also necessary to analyze the structural deformation displacement of the wing, as shown in Figure 10. It can be seen that as the angle of attack increases, the global deformation of the wing, as well as the displacement deformation of the leading edge and trailing edge of the wing tip, all increase. However, the displacement deformation of the leading edge of the wing tip is relatively small, with a maximum deformation of only 4.75%. This confirms that the leading edge reinforced hybrid membrane wing has good resistance to spanwise bending deformation. From the displacement contour, it can be seen that the deformation form of the wing is consistent, and the global maximum deforma-

tion comes from the deformation of the elastic membrane. The deformation of the elastic membrane is mainly caused by an upward protrusion in the middle, gradually increasing from the wing root to the wing tip. The small displacement deformation of the leading edge of the wing tip indicates that the hybrid membrane wing with strengthened leading edge has good resistance to spanwise bending deformation. The displacement deformation of the trailing edge of the wing tip is greater than that of the leading edge of the wing tip. The main reason is that the trailing edge of the wing tip is an arch-shaped flat wing and the Young's modulus of the elastic film is small, which reduces the stiffness of the trailing edge, but the deformation is small, indicating good resistance to torsional deformation.

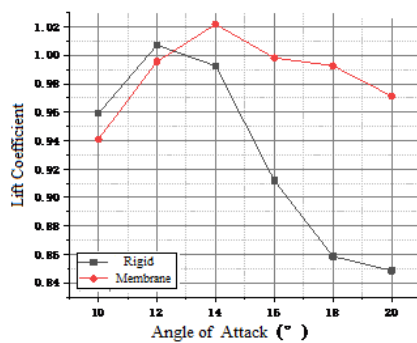


Figure 5: Lift coefficient varying with the angle of attack of the wing.

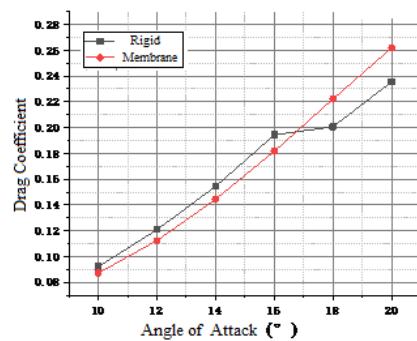


Figure 6: Drag coefficient varying with the angle of attack of the wing.

5 CONCLUSION

This paper proposes a leading edge reinforced membrane wing to address the issue of low spanwise stiffness and susceptibility to significant spanwise bending and torsion of membrane wings. The aerodynamic model and structural

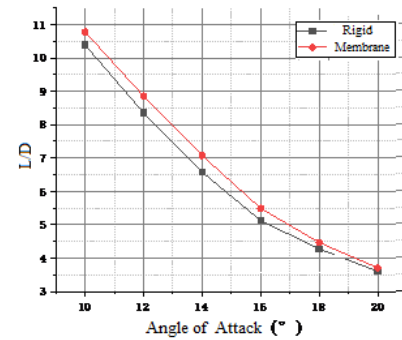


Figure 7: Ratio of the lift to drag varying with the angle of attack of the wing.

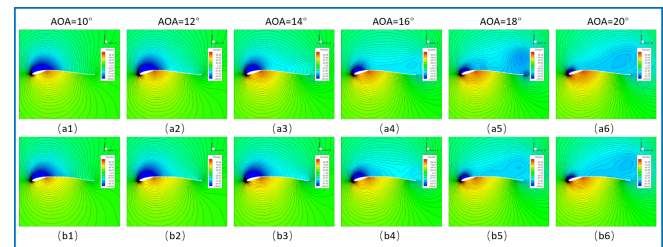


Figure 8: Pressure contour at the cross section of the wing.

model of the leading edge strengthened hybrid membrane wing were established. Based on the two-way fluid structure coupling model of the leading edge strengthened hybrid membrane wing, the aerodynamic and structural deformation analysis was carried out, and compared with the rigid wing, the results showed that the lift coefficient of the membrane wing before stall was less than that of the rigid wing, but the lift coefficient after stall was greater than that of the rigid wing, and the lift drag ratio of the membrane wing was greater than that of the rigid wing throughout the whole process, with higher aerodynamic efficiency. Analysis of struc-

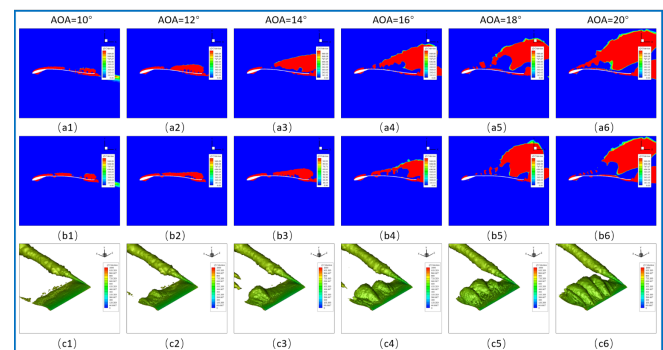


Figure 9: Q-criterion vorticity contour around the wing.

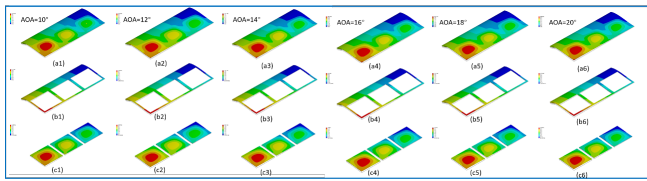


Figure 10: Structural deformation displacement contour of the wing.

tural displacement deformation found that the displacement deformation of the leading edge of the wing tip was relatively small, and the deformation of the wing mainly occurred in the elastic membrane part, verifying that the hybrid membrane wing with strengthened leading edge has good resistance to spanwise bending deformation. The results indicate that the leading edge reinforced hybrid membrane wing has good resistance to spanwise bending deformation, while retaining the adaptive passive deformation ability of traditional membrane wings, and has good stall performance.

In the further research work, it shall carry on wind tunnel tests to verify the simulation results.

ACKNOWLEDGEMENTS

The authors would like to thank the HPC center of the Xi'an Jiaotong University for their help on the simulation.

REFERENCES

- [1] B. BEGUIN, C. BREITSAMTER, and N. ADAMS. Aerodynamic investigations of a morphing membrane wing. *AIAA Journal*, 50(11):2588–2599, 2012.
- [2] M. D. Maughmer. *A Comparison of the Aerodynamic Characteristics of Eight Sailwing Airfoil Sections*. Princeton Univ, LLC, New York, NY, 1979.
- [3] K. Fujita, K. Takahashi, and H.C. NAGAI. Permeable membrane wing characteristics at reynolds number order of 10,000. In *AIAA Scitech 2021 Forum*, 2021.
- [4] R. M. WALDMAN and K. S. BREUER. Camber and aerodynamic performance of compliant membrane wing. *Journal of Fluids and Structures*, 68(1):390–402, 2017.
- [5] A. SONG, X. TIAN, and E. ISRAELI. Aeromechanics of membrane wings with implications for animal flight. *AIAA Journal*, 46(8):2096–2106, 2008.
- [6] A. Pelletier and T. J. Mueller. Low reynolds number aerodynamics of low-aspect ratio, thin/flat/cambered-plate wings. *Journal of Aircraft*, 37(5):825–832, 2008.
- [7] G. E. Torres and T. J. Mueller. Low-aspect ration wing aerodynamics at low reynolds numbers. *AIAA Journal*, 45(5):865–873, 2004.

- [8] R. ALBERTANI, B. STANFORD, and J. P. HUBNER. Aerodynamic coefficients and deformation measurements on flexible micro air vehicle wings. *Experimental Mechanics*, 47(5):625–635, 2007.

Measuring Disturbance Response of a Fully-Actuated UAV using Surging Flow Wind Tunnel Experiments

J. Chen*, N.J. Kay and K.A. Stol

Department of Mechanical and Mechatronics Engineering, University of Auckland, New Zealand

ABSTRACT

This paper presents a novel experimental setup in a wind tunnel that is capable of measuring the frequency response of an over-actuated Unmanned Aerial Vehicle (UAV). The setup generates wind surges with a sinusoidal time response, allowing for free-flight testing of the UAV at various frequencies. Importantly, the surging flow method employed in this study offers a predictable and repeatable experiment compared to other wind tunnel tests that utilised turbulence grids. The closed-loop frequency response of the UAV to wind disturbances is plotted, revealing peaks in the gain response that are not present in simulated results. These unexpected peaks identify that system dynamics that are not modelled in the simulation are present in the physical system.

These findings demonstrate the effectiveness of the experimental setup and provide valuable insights into the UAV's response to disturbances.

1 INTRODUCTION

Unmanned aerial vehicles (UAVs) have experienced a large growth in popularity in the past few decades in both commercial and academic sectors. With the miniaturisation of sensors and embedded computers, small multirotor UAVs in particular are becoming a widely used instrument in a variety of fields such as surveillance, mapping, agriculture and delivery due to features such as their agility and relatively moderate production costs [1, 2, 3].

Conventional UAV designs have all rotors applying force in a parallel direction to counteract gravity and thus under-actuated dynamics due to the coupling between the horizontal translational and rotational dynamics. Fully-actuated UAVs are a type of UAV that have an equal number of control inputs to the number of degrees of freedom (DoF), and each DoF is decoupled and can thus be controlled by an independent control input. This class of UAVs is able to apply forces in a plane perpendicular to the rotor axis making them more suitable than conventional designs for certain tasks, examples being aerial physical interaction and wind gust rejection [4, 5].

Over-actuated UAVs are another type of UAV that share similar characteristics to fully-actuated UAVs with added redundant control inputs, allowing for greater fault tolerance and robustness.

The optimal rotor configuration for fully-actuated UAV designs is often application-dependent, where different configurations are more suitable for certain design goals, such as better dynamic maneuverability, minimum energy consumption, greater force bandwidth or wrench generation [3]. Different control algorithms have also been developed throughout literature to utilise the advantages of these configurations [6, 7]. Examples of small fully-actuated and over-actuated multirotor UAVs have been developed by several researchers, demonstrating their potential for high levels of maneuverability [8, 9, 10, 11].

With the increased popularity of UAVs, there is an increased need for accurate flight performance while resisting external disturbances due to factors such as wind, aerodynamic ground effects or when performing aerial manipulation [12]. Accurate modeling of the dynamics of these UAVs is thus crucial for improving their performance through better design and control. Many research groups have made use of various analytical and empirical methods to derive UAV aerodynamics [13, 14, 15], which can then be used to develop suitable simulation models for controller design and testing. Modelling has been shown to be a useful tool for understanding the behavior of multirotor UAVs but may not capture all of its complexities accurately due to unmodelled dynamics. Experimental methods are then used to validate and refine these UAV models.

Free-flight testing is a widely used approach in UAV literature to assess the vehicle's dynamics and performance in real-world environments [16, 17, 18]. This method involves flying the UAV under various conditions, and comparing the recorded data with simulation results to evaluate the aircraft's performance. Outdoor testing provides the most realistic representation of real-world conditions but introduces uncertainties and limited control over external factors. Indoor testing offers controlled conditions but are limited by space restrictions.

Wind tunnel testing, on the other hand, provides controlled experimental conditions for UAV flight. Although the wind tunnel environment does not fully replicate the real-world wind conditions and turbulence experienced during outdoors flight, its predictable nature offers distinct advantages for precise and repeatable assessment of the stability

*Email address(es): jche825@aucklanduni.ac.nz

and control of the vehicle. Additionally, the use of motion capture systems in the wind tunnel setting enables highly accurate data collection, which may be challenging to achieve or cost-prohibitive in outdoor environments. The effectiveness of wind tunnel testing has been demonstrated in examples such as [19, 20], in which testing was performed using either constant wind, or turbulent wind which contains a broad spectrum of irregular wind speeds.

In past literature, it was noted that some fully-actuated drones have an advantage over under-actuated drones in terms of actuator bandwidth or rise time [9, 21]. The irregularity of turbulent wind was found to be unsuitable for properly evaluating the effects of this advantage. To investigate this advantage, equipment was built by Buchanan [22] to generate predictable sinusoidal wind 'surges' which are described in [23] and hereon referred to as 'surging flow'. To the author's knowledge, UAV free-flight testing has yet to be performed under the conditions described. This paper therefore extends work done in [20] by investigating the frequency response of a fully-actuated UAV in free-flight using surging flow conditions generated by a wind tunnel.

The paper is organised as follows. In Section 2, the fully-actuated UAV and testing environment are introduced. The experimental process and results are then discussed in Section 3, and finally the conclusions drawn and future work are summarised in Section 4.

2 SETUP DESCRIPTION

2.1 UAV Airframe and Controller

This section introduces the UAV investigated in this paper, known as the canted-rotor planar octorotor (CRPO) UAV, which was previously designed in research conducted at the university [20]. As the CRPO has 8 control inputs for 6 DoF of motion, it is technically an over-actuated UAV, however it is treated as a fully-actuated UAV as this characteristic is not explored in this paper.

Two different reference frames are used in this paper. One is about the *body* of the aircraft, denoted as $B[x, y, z]$ about the forward, right and down directions of the aircraft respectively. Additionally, euler angles for roll ψ , pitch θ and yaw ϕ of the aircraft are positive clockwise about these three respective body axes. The *world* coordinate frame then describes the environment the aircraft is in $W[x, y, z]$ where the W_{xy} -plane is parallel to the ground, W_z is positive upwards normal to the ground, and centred on the ground in the flight zone described later in Section 2.2.

The CRPO is similar to a conventional planar octorotor UAV, except the rotors of the CRPO have been canted such that adjacent propellers are tilted along the B_x - and B_y -axes at a fixed angle ζ in opposing pairs, as shown in Figure 1. This allows for decoupling of position and attitude control, where increasing thrust from certain rotors while decreasing thrust from others generates a net thrust along its B_{xy} -plane without generating moments or affecting the net force normal

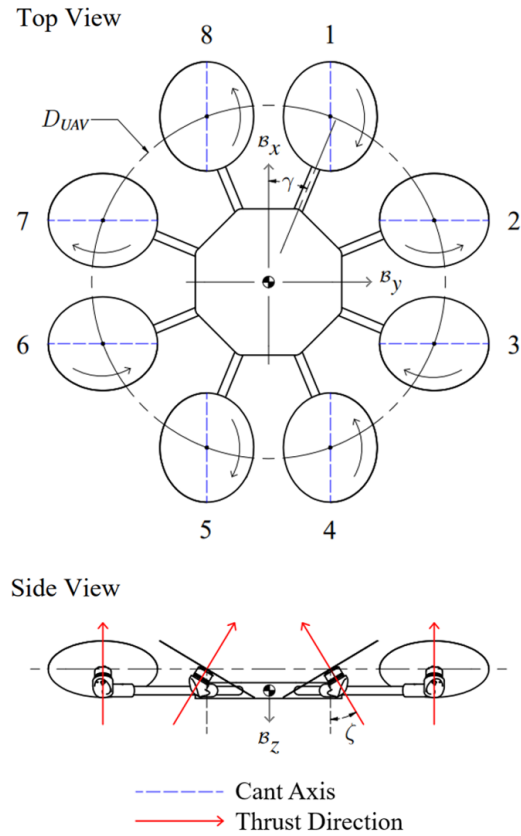


Figure 1: Diagram of the canted-rotor planar octorotor UAV

to this plane in the B_z direction.

The CRPO is able to produce thrust in the world W_x - and W_y -axes using two different methods: directly via body thrust as described above, or in a manner similar to an under-actuated UAV by first tilting the aircraft so the B_z -axis thrust has components in the W_x - or W_y -axes. These two actuation methods will hereon be described as *vectored thrust* and *attitude-based thrust* respectively. Experiments showed that the bandwidth of vectored thrust compared to attitude-based thrust was 8Hz and 1.7Hz respectively, suggesting vectored thrust was better suited for counteracting high frequency disturbances. However, attitude-based thrust was determined to be capable of producing a higher magnitude of force in the W_{xy} -plane.

The CRPO UAV model was primarily constructed using carbon fibre plates and tubes. It is controlled by a Holybro Pixhawk 4 flight controller running PX4 Autopilot. The CRPO is equipped with a radio module for manual operation via a radio controller, as well as a WiFi module for communication with a ground control station (GCS) laptop using the MAVLink protocol. The aircraft has a rotor-to-rotor diameter D_{UAV} of 0.5m and a flight mass of 1.72kg. An image of the aircraft is provided in Figure 2.

The CRPO can be operated with two different position



Figure 2: Photo of the canted-rotor planar octorotor UAV

controllers: a baseline PX4 controller based on PID, which does not utilize vectored thrust, or an H_∞ controller designed by Bannwarth [20]. The H_∞ controller is a linear multivariable control system, which was optimised to mitigate disturbances based on a linearised model of the aircraft constructed at a station-keeping operating point in a 5.6m/s headwind. The controller takes advantage of the strengths of the CRPO's separate actuation methods by individually penalising inputs and outputs based on frequency. This approach is thus able to leverage the higher frequency bandwidth enabled by vectored thrust as well as the greater magnitude of thrust provided through attitude-based thrust.

The design features of the CRPO and a system of equations representing its dynamics and aerodynamics are described in greater detail by Chen and Bannwarth et al. in [21, 24]. Bannwarth also describes how the system of equations were then used to construct a simulation model of the plant and controller in MATLAB Simulink. The structure of this simulation model is shown in Figure 3 and it is used later in Section 3 to generate simulated results for comparison with the experimental data.

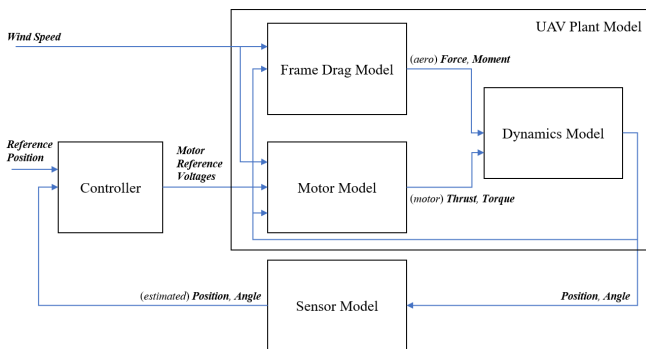


Figure 3: CRPO Simulation Model

2.2 Surging Flow System

The flight environment used in this experiment to generate surging flow is the University of Auckland's Boundary Layer Wind Tunnel (BLWT). Detailed descriptions for the wind tunnel and modifications made in the flight zone to

enable free flight testing and surging flow can be found in [15, 20, 22, 25]. This section will briefly summarise its key features in relation to the free-flight tests that were conducted in this paper.

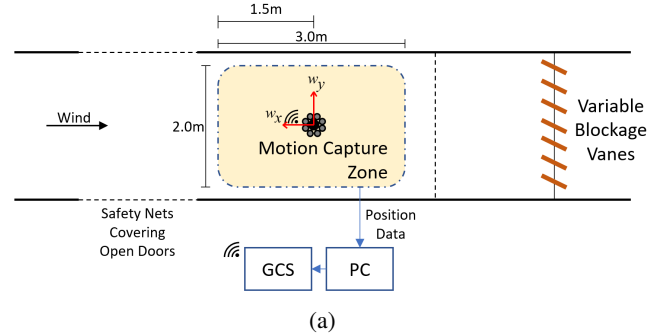


Figure 4: a) Diagram of BLWT flight zone and b) photo of BLWT flight zone taken from upstream

The BLWT is a closed-loop wind tunnel with a 20m test section that is 3.6m in width and 2.5m in height. Within this test section there is a much smaller motion capture zone suitable for free-flight experiments that is 2.0m in width and 3.0m in length. Figure 4a shows the layout of the flight environment for this experiment within the test section of the wind tunnel. The wind tunnel is able to generate wind speeds within the tunnel of up to $20m.s^{-1}$. This has been restricted to a maximum of $10m.s^{-1}$ for safety reasons in the experiments conducted due to the flow blockage caused by the variable blockage vanes. Wind speed in the tunnel is tracked in two locations both upstream and downstream of the test area by pitot tubes.

The variable blockage vanes are a set of seven motorised vanes, shown as the vertical wooden slats downstream of the tunnel in Figure 4b. The angle of these vanes can be adjusted from an open angle of 0° shown in the Figure to a closed angle of 90° to increase blockage within the tunnel and thus de-

crease wind speed in the testing zone separately from the primary wind tunnel fan. To reduce pressure loading on the wind tunnel fans, the tunnel side doors must be left open when the vanes are operating and thus nets have been placed across the doors and behind the flight zone for safety.

The vanes are able to repeatedly generate discrete one-dimensional (1-D) gusts of wind in the $\mathcal{W}_x$ -axis with minimal turbulence. This can be controlled to generate a 'surging flow', or a set of gusts with speeds that resemble a 1-D sinusoidal signal in the time domain, and thus a single frequency component. The frequency of these wind gusts is determined by the frequency the vanes are operating and the gust amplitude is related to the turning angle of the vane, e.g. a surging flow may be generated using sweeps between only 0° to 30° . Characterisation of the wind generated by this system has been conducted by Kay et al. in past research and can be found in [23].

The motion capture zone shown in Figure 4 provides accurate position and attitude data with sub-millimetre position accuracy. Motion capture is performed by twelve Opti-Track Flex 3 cameras, shown embedded in the roof in Figure 4b, which track circular reflective markers on the object of interest. Several of these markers can be seen attached to the body of the CRPO in Figure 2.

The localisation of position data is achieved by processing the camera feeds on a PC, which transmits the data to a ground control station (GCS) laptop. The GCS laptop runs the MAVROS package and establishes a connection to the UAV through a Wi-Fi module. By utilising this Wi-Fi connection, the position data is transmitted to the UAV flight controller using MAVLink messages. On the UAV, the received position data is filtered with on-board sensor data, thus minimising errors associated with sensor inaccuracies.

3 SURGING FLOW EXPERIMENTS

During these station-keeping experiments, power was supplied to the CRPO by a power supply via tether, allowing for greater flight endurance. The flight mass mentioned at the end of Section 2.1 includes the mass of this tether and the power supply Voltage is set to 12.6V to simulate a 3S Lithium-Polymer battery.

To maintain consistency between experiments, it is desirable that the mean wind speed and amplitude are constant for all surging flow frequencies. However, when the variable blockage vanes described in Section 2.2 are in operation the mean wind speed decreases significantly from the open-vane wind speed; this drop changes with both surging flow frequency and vane angle. Similarly, the amplitude of wind gusts is also affected by both maximum vane turn angle and frequency. For example, to achieve wind speeds with a mean, amplitude and frequency of 5.6ms^{-1} , 0.56ms^{-1} and 1.0s^{-1} (Hz) respectively, an open vane wind speed of 8.3ms^{-1} was needed and vanes were swept from 0° to 67° . As such, appropriate open-vane wind speeds and vane turn angles must



Figure 5: CRPO station-keeping in the BLWT under surging flow. Wind sock visually indicates wind direction.

be selected before conducting free-flight experiments to generate a consistent set of winds for use in said experiments.

The H_∞ controller described at the end of Section 2.1 is linearised about a mean wind speed of 5.6ms^{-1} , which was the target mean wind speed for free-flight experiments using a turbulence grid conducted by Bannwarth [20]. As such, this wind speed $U_{mean} = 5.6\text{ms}^{-1}$ was again chosen to be the mean wind speed for the surging flow experiments conducted in this paper. To remain within the linearised region, wind speed amplitude U_{amp} was then set to a small value; this was chosen as 10% of the mean wind speed or 0.56ms^{-1} . Experiments were then conducted across a range of frequencies $f = \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 1.0\}$. Experiments were also repeated using both a baseline PX4 controller based off of PID and the H_∞ controller. To ensure the desired mean and amplitude wind speeds at each frequency, interpolation was applied to wind profiling data collected by Kay et al. [23] and confirmed experimentally using a Cobra probe 3-axis wind sensor placed at the station-keeping position of the CRPO in the experiment, described later in Section 3.1. During this stage, wind profile logs were also collected, providing data on wind speeds within the flight zone in relation to the variable vane turn angle.

3.1 Experiment Procedure

During the experiment the CRPO is set to *offboard*-mode, which is an autonomous mode where the UAV receives a constant stream of position setpoints for the desired hover position, $\mathcal{W}_{x_{ref}} = [0 \ 0 \ 1.25]^\top \text{m}$. This corresponds to a location 1.25m above the centre of the motion capture area, aligned with the centre of the wind tunnel and is the position the UAV will attempt to hold during the experiment. The H_∞ controller is turned on here if appropriate for the experiment.

Once the wind tunnel is activated, the wind speed is gradually increased to the desired open-vane wind speed. Sufficient time is allowed for the wind speed to reach a steady state within the tunnel. Subsequently, the variable-blockage vanes

are activated, and their operation is sustained for a duration of 40 seconds to 2 minutes, depending on the test frequency. This ensures that the surging flow within the tunnel achieves a steady state for a meaningful period. A photo taken during one test flight can be seen in Figure 5.

During the experiment flight logs were recorded, capturing aircraft position, attitude, and control allocation data. These logs were subsequently processed to extract flight data aligned with the variable vane turn angle. A sample from one experiment can be seen in Figure 6, after synchronising time between the aircraft log and profiled wind data. Curve fitting tools have been used to approximate the shape of the responses and most importantly the amplitudes of each response, which are then used to generate the plots discussed in Section 3.2.

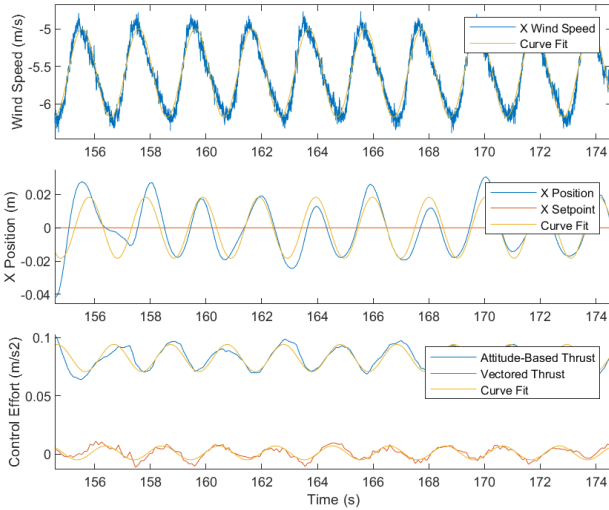


Figure 6: Combined logs from CRPO station-keeping in the BLWT using the H- ∞ controller under 0.5Hz surging flow. From the top, graphs represent estimated wind speed, UAV position and UAV control effort in the w_x -axis

3.2 Experiment Results

Results for each experiment were recollected by using the wind profile logs collected as inputs to the simulation model described in Figure 3. The closed-loop frequency response to wind disturbance from both simulation and experimental results for both baseline and H- ∞ controllers is then shown in Figure 7. In this graph, "Position Gain" represents the difference in decibels between the output position amplitude when fitted to a sinusoidal curve and the input wind speed amplitude when fitted to a sinusoidal curve at each frequency. Although not all signals exhibited perfect sinusoidal behavior, especially at lower frequencies with minimal position deviation, the analysis revealed that sinusoidal curve fits yielded satisfactory results. Specifically, the simulated results consistently achieved an R-squared value above 0.8, indicating a

strong fit to the sinusoidal curve. Similarly, the experimental results exhibited an R-squared value above 0.7, indicating an acceptable level of accuracy for the curve fits. Despite some deviations from ideal sinusoidal patterns, the overall quality of the fits suggests their suitability for the analysis and interpretation of the data.

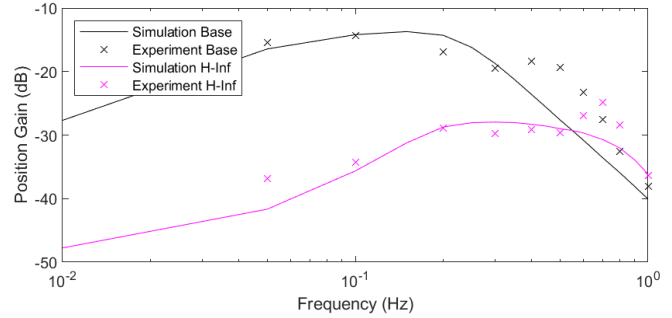


Figure 7: Closed-loop frequency response of position error gain due to wind disturbance

The simulated results appear to represent the performance of the real system in free-flight experiments fairly accurately except for around the 0.4Hz and the 0.7Hz region for baseline and H- ∞ controllers respectively where there appears to be an additional peak in the results from free-flight experiments. It is likely that these peaks were caused by unmodelled or inaccurately modelled system dynamics, such as the influence of the power tether on the UAV's mass moment of inertia I in the experiment. Furthermore, the variations in damping effects induced by the different controllers may contribute to the disparate frequencies at which these peaks occur.

Additionally, it was previously difficult to experimentally confirm whether the frequency-based control allocation of the H- ∞ controller described in Section 2.1 was effective at mitigating position gain. By analysing control allocation at each frequency, Figure 8 demonstrates that in relation to position error amplitude, control effort of the vectored thrust appears to increase more significantly than attitude-based thrust with surging flow frequency.

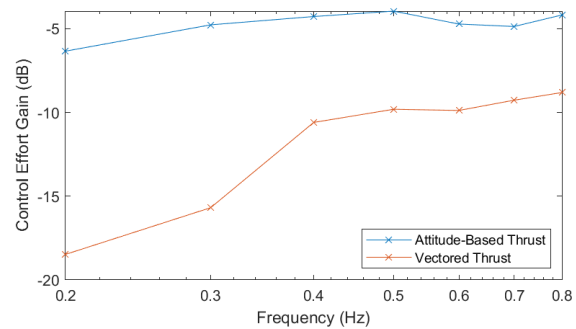


Figure 8: Closed-loop frequency response of control effort gain due to position error of the H- ∞ controller

4 CONCLUSION

In this study, we have introduced a novel experimental setup in a wind tunnel to measure the disturbance response of a fully-actuated UAV. By generating wind surges with a sinusoidal shape, we conducted free-flight tests across a range of winds with a single frequency component, enabling the analysis of the UAV's closed-loop frequency response to wind disturbances.

We observed an unexpected peak in the closed-loop frequency response of the position error of the CRPO found through the surging flow wind tunnel experiments that wasn't present in simulated experiments. This difference identifies the effect that the unmodelled dynamics in the physical system, such as the effects of the power supply tether or frame vibrations, has on results from experiments using the CRPO. Additionally, this apparatus was used to investigate the effect of frequency-weighted control of the two separate actuation methods for horizontal thrust of the CRPO when using the H_∞ controller, and the results aligned well with expectations for the performance of its frequency weighting.

These findings highlight the usefulness of this experimental setup for identifying unmodelled dynamics and investigating control strategies of over-actuated UAVs.

4.1 Future Work

Further investigation could be performed on several areas identified by this work. Firstly, the sources of the unexpected peaks in the frequency response should be examined, and the unmodelled dynamics should be identified to further refine the simulation model and controller. Surging flow experiments could then be used to validate the accuracy of these changes.

Furthermore, the frequency response analysis of the UAV's dynamics offers valuable insights for refinement of the UAV's design and control system. For the case of the H_∞ controller, by examining performance across different frequencies, a more effective frequency weighting may be applied to reduce gain at specific frequencies and thus improve overall performance. Surging flow experiments could then be used to investigate the effectiveness of these changes at specific surging flow frequencies.

ACKNOWLEDGMENTS

The research reported in this article was conducted as part of "Enabling unmanned aerial vehicles (UAVs) to use tools in complex dynamic environments UOCX2104", which is funded by the New Zealand Ministry of Business, Innovation and Employment.

REFERENCES

- [1] F. Nex, C. Armenakis, M. Cramer, D.A. Cucci, M. Gerke, E. Honkavaara, A. Kukko, C. Persello, and J. Skaloud. Uav in the advent of the twenties: Where we stand and what is next. *ISPRS Journal of Photogrammetry and Remote Sensing*, 184:215–242, 2022.
- [2] Bas Vergouw, Huub Nagel, Geert Bondt, and Bart Custers. *Drone Technology: Types, Payloads, Applications, Frequency Spectrum Issues and Future Developments*, pages 21–45. T.M.C. Asser Press, The Hague, 2016.
- [3] S. Suzuki. Recent researches on innovative drone technologies in robotics field. *Advanced Robotics*, 32(19):1008–1022, 2018.
- [4] Yuichi Tadokoro, Tatsuya Ibuki, and Mitsuji Sampei. Classification and structural evaluation of fully-actuated hexrotor uavs. In *2018 Annual American Control Conference (ACC)*, pages 1945–1950. IEEE, 2018.
- [5] Ramy Rashad, Jelmer Goerres, Ronald Aarts, Johan B. C. Engelen, and Stefano Stramigioli. Fully actuated multirotor uavs: A literature review. *IEEE Robotics & Automation Magazine*, 27(3):97–107, 2020.
- [6] Jinho Kim, S. Andrew Gadsden, and Stephen A. Wilkerson. A comprehensive survey of control strategies for autonomous quadrotors. *Canadian Journal of Electrical and Computer Engineering*, 43(1):3–16, 2020.
- [7] Sherif I. Abdelmaksoud, Musa Mailah, and Ayman M. Abdallah. Control strategies and novel techniques for autonomous rotorcraft unmanned aerial vehicles: A review. *IEEE Access*, 8:195142–195169, 2020.
- [8] Z. Jeremy Chen, Karl A. Stol, and Peter J. Richards. Preliminary design of multirotor uavs with tilted-rotors for improved disturbance rejection capability. *Aerospace Science and Technology*, 92:635–643, 2019.
- [9] Guangying Jiang and Richard Voyles. A nonparallel hexrotor uav with faster response to disturbances for precision position keeping. In *2014 IEEE International Symposium on Safety, Security, and Rescue Robotics (2014)*, pages 1–5, 2014.
- [10] Sijia Wang, Lei Ma, Binbin Li, and Kai Zhang. Architecture design and flight control of a novel octopus shaped multirotor vehicle. *IEEE Robotics and Automation Letters*, 7(1):311–317, 2022.
- [11] Florentin von Frankenberg and Scott Nokleby. Disturbance rejection in multi-rotor unmanned aerial vehicles using a novel rotor geometry. In *Proc. 4th Int. Conf. Control Dynamic Systems and Robotics*, 2017.
- [12] Pedro Sanchez-Cuevas, Guillermo Heredia, and Anibal Ollero. *Multirotor Aerodynamic Effects in Aerial Manipulation*, pages 67–82. Springer International Publishing, Cham, 2019.

- [13] Nedim Osmić, Muhamed Kurić, and Ivan Petrović. Detailed octorotor modeling and pd control. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 002182–002189, 2016.
- [14] Praveen Abbaraju, Xin Ma, Guangying Jiang, Mo Rastgaar, and Richard M. Voyles. Aerodynamic modeling of fully-actuated multirotor uavs with nonparallel actuators. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9639–9645, 2021.
- [15] Jérémie X. J. Bannwarth, Z. Jeremy Chen, K. A. Stol, B. A. MacDonald, and P. J. Richards. Development of a wind tunnel experimental setup for testing multirotor unmanned aerial vehicles in turbulent conditions. In *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 724–729, 2018.
- [16] Nicolas Michel, Peng Wei, Zhaodan Kong, Anish Kumar Sinha, and Xinfan Lin. Modeling and validation of electric multirotor unmanned aerial vehicle system energy dynamics. *eTransportation*, 12:100173, 2022.
- [17] Christina Ivler, Robert Niemiec, Farhan Gandhi, and Frank C Sanders. Multirotor electric aerial vehicle model validation with flight data: Physics-based and system identification models. In *Vertical Flight Society 75th Annual Forum, Philadelphia, PA, USA*, 2019.
- [18] He Zhu, Hong Nie, Limao Zhang, Xiaohui Wei, and Ming Zhang. Design and assessment of octocopter drones with improved aerodynamic efficiency and performance. *Aerospace Science and Technology*, 106:106206, 2020.
- [19] Fu Junquan, Gong Zheng, Shi Zhiwei, Zhang Tongren, and Zhu Jiachen. Transition controllability research on tri-tilt rotor unmanned aerial vehicle in wind tunnel free flight tests. In *2017 9th International Conference on Modelling, Identification and Control (ICMIC)*, pages 527–532, 2017.
- [20] Jérémie X. J. Bannwarth. *Aerodynamic Modelling and Wind Disturbance Rejection of Multirotor Unmanned Aerial Vehicles*. PhD thesis, University of Auckland, Auckland, New Zealand, 2021.
- [21] Z. Jeremy Chen, Jérémie X. J. Bannwarth, Karl A. Stol, and Peter J. Richards. Analysis of a multirotor uav with tilted-rotors for the purposes of disturbance rejection. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 864–873, 2018.
- [22] Rory D. Buchanan. Aerodynamic modelling and rejection of wind disturbances for multirotor unmanned aerial vehicles. Master’s thesis, University of Auckland, Auckland, New Zealand, 2020.
- [23] Nicholas J. Kay, Rory D. Buchanan, Peter J. Richards, and Karl A. Stol. Generating sinusoidal flow surges in a large closed-loop wind tunnel. In *16th International Conference on Wind Engineering*, to appear.
- [24] Jérémie X. J. Bannwarth, Z. Jeremy Chen, Karl A. Stol, Bruce A. MacDonald, and Peter J. Richards. Aerodynamic force modeling of multirotor unmanned aerial vehicles. *AIAA Journal*, 57(3):1250–1259, 2019.
- [25] Richard G. J. Flay and Yin Fai Li. New zealand country report 2021. *Construction*, 2:65–77, December 2022.

Multi Method Local Path Planning Considering Flight Dynamics Constraints for Collision Avoidance

L. Danielmeier*, S. Seitz, M. Hartmann, N. Voget and D. Moormann
RWTH Aachen University, Wüllnerstraße 7, Aachen, Deutschland

ABSTRACT

The regulation for automatic UAV flights in Europe requires a UAV to be able to avoid collisions. Ideally, the avoidance path of the UAV should be explainable for pilots flying the aircraft to be avoided. However, accomplishing this using only one avoidance method can be difficult. Methods that generate a path that is understandable by humans are often not capable of solving complex avoidance scenarios. On the other hand, using a complex avoidance method for every avoidance scenario can result in unexplainable avoidance paths for simple avoidance scenarios. Therefore, this paper presents a local path planning concept for collision avoidance, using four different path planning methods in combination with an avoidance method selection logic. These are vertical avoidance, avoidance by deviating to a path that is parallel to the original flight path, varying the speed on the original path, and the mCOWEX algorithm. This yields simple and understandable avoidance paths whenever possible, while at the same time being able to fall back to the high success rate of the more complex methods.

1 INTRODUCTION

The demand for UAV capable of fully automatic BVLOS operation is steadily increasing due to their versatility in many different use cases such as transport of goods or search and rescue missions. For legal BVLOS operation of automatic UAVs in the EU a fully automatic, robust, but also deterministic collision avoidance is required. In this context, deterministic means, that given a similar avoidance situation, i.e. two slightly different head-on collisions, the UAV should still use a similar avoidance path for both evasions, as a human operator would. This is required, so that a human pilot encountering the UAV as a collision partner, is able to predict what path the UAV will take based on the collision scenario. For some approaches to automatic collision avoidance, such as A* [1], COWEX [2], mCOWEX [3] or potential field methods [4] this poses a problem, as the exact same avoidance scenario will generate the exact

same avoidance path, but a slightly different scenario might yield a vastly different path. The difference might be as small as a meter offset in the intruder position relative to the UAV. This is problematic for the explainability of the avoidance maneuver, as a human would be unable to predict the avoidance path the UAV might take for a given scenario. On the other hand, the avoidance method cannot be so simple, that it will not be able to find a collision free path in a more complex avoidance scenario.

For an avoidance path to be successful, it must not only be free of collisions, but it must also be flyable with respect to the flight dynamics constraints, such as minimal turn radius, acceleration and deceleration capability as well as maximum climb and decent rates. There are two approaches to consider these flight dynamics constraints of the aircraft for avoidance path planning. In the first approach, algorithms that do not consider the constraints are used to plan a number of avoidance paths. These are checked for flight dynamics compliance after they are generated and the shortest path that complies to the flight dynamics constraints is used while those who do not comply are discarded. This is not computationally efficient, as the available computing time is used for paths that are not flyable and have to be discarded. In other approaches, methods and algorithms that consider the flight dynamics constraints during planning such as constrained A* [5], constrained RRT [6] or mCOWEX are used. These algorithms have a higher computation time in comparison to unconstrained algorithms, but only have to compute one successful path. Thus the available computation time is used more efficiently.

Therefore, this paper proposes a multi method local path planner to generate explainable avoidance paths for simple situations using simple avoidance methods, while being capable of solving complex avoidance scenarios using the more complex mCOWEX algorithm.

First, the four collision avoidance methods used in this paper are explained. Second, the logic, with which the path generation method is chosen based on the avoidance scenario is presented. Third, Monte Carlo simulations for different sets of avoidance scenarios are conducted with each method separately as well as the combination of all methods i.e. the multi method local path planner. The results are then used to compare the success rates of each standalone method with the success rate of the multi method local path planner.

*Email address: danielmeier@fsd.rwth-aachen.de

Thereafter, examples of avoidance paths generated using each of the methods of the multi method local path planner will be shown. The results presented in this section will also be briefly discussed.

2 COLLISION AVOIDANCE METHODS

This section will give an overview of the four collision avoidance methods used in the multi method local path planner presented in this paper. As these methods are path planning algorithms they all rely on an estimation of the intruder position at a given time. This position is estimated using a 99% confidence interval of the possible positions of the intruder given by the covariance matrices of an IMM Kalman Filter, i.e. [7]. This area of possible intruder positions is displayed by the dark red ellipse i.e. Fig. 7. The light red ellipse around the area of possible intruder positions is a safety zone of a pre-defined size. This safety zone extends to 300m horizontally and ± 50 m vertically. The safety zone for static obstacles is displayed by a light red circle around the dark red obstacle. The UAV's position, speed and heading on a given path are calculated by the UAV's flight guidance system, therefore these are also assumed to be known [3]. During flight a collision detection system continuously monitors the information about intruders, supplied by a different systems such as FLARM and ADS-B. A collision is defined as the loss of a defined minimum separation between any obstacle or any intruder and the UAV. If a possible collision has been detected, the local path planner is triggered and all four path planning algorithms are executed in parallel. Thereafter, all valid paths are stored and the avoidance method selection logic is used to select a path. This selection process will be explained in Sec. 3.

2.1 Vertical Avoidance

The first method of collision avoidance used in this paper alters the height profile of the original path to avoid collisions. This offers the advantage that the geofence and other static 2D obstacles, that were considered in the original path, do not have to be considered while generating the avoidance path. A disadvantage of altering the height profile, is that information about the ground height of the surrounding terrain has to be considered and processed for the avoidance path generation. To generate a collision free height profile, the times t_{enter} and t_{exit} and the positions p_{enter} and p_{exit} at which the UAV enters and leaves the safety zone of the intruder are calculated. Then, the maximum and minimum heights of the intruder's vertical safety zone for the duration between entry and exit are determined. Using these parameters, the altitudes h_{climb} and h_{descent} required for a collision free height profile are derived, see. Fig. 1. To guarantee, that no airspace infringements or ground collisions occur, these altitudes are then checked for compliance with the ground profile, as well as all airspaces crossed by the flightpath. If there is a conflict with the ground or the airspace, the maneuver can not be executed.

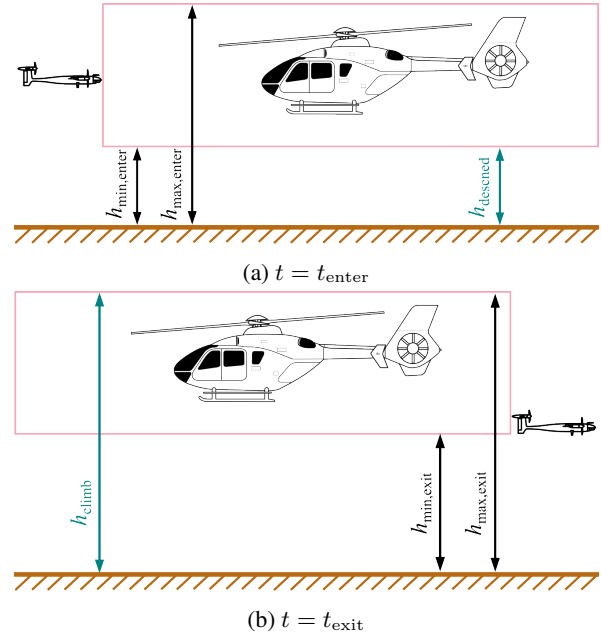


Figure 1: Visualization of the altitudes h_{climb} and h_{descent} at $t = t_{\text{enter}}$ and $t = t_{\text{exit}}$.

The following steps are the same for the climb and the descent maneuver, therefore they are only explained for the climb maneuver.

First, the point on the original path, where the climb must begin in order to reach h_{climb} two seconds before t_{enter} is calculated using the maximum climb rate of the UAV. If the UAV has already passed this point the maneuver is not fly-able and therefore either the descent maneuver or a different approach must be used. After the climb, altitude is held at h_{climb} from $t_{\text{enter}} - 2\text{ s}$ to $t_{\text{exit}} + 2\text{ s}$ and thereafter the UAV descends with maximum descent rate until the height profile of the original path is reached.

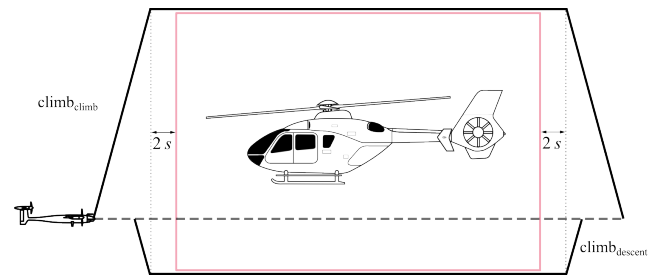


Figure 2: Visualization of climbs required for the climb and the descent maneuver.

If both the climb and descent maneuver are possible, the required climb times for both maneuvers are compared. As climbing is more energy consuming than descending, and both maneuvers can have a climb and a descent, the maneuver with the smaller climb is chosen. This is shown in Fig. 2,

where the required climb for the climb maneuver is much larger than the required climb for the descent maneuver.

2.2 Avoidance using Parallel Paths

Within the second method a flight path parallel to the original flight path is used to generate a collision free avoidance path. The first step for generating a parallel avoidance path is to determine to which side the path should deviate from the original path. This is done by analyzing the avoidance scenario and then choosing a side to avoid the collision according to SERA.3210, the Standardized European Rules of the Air [8]. These rules define how a potential conflict between two aircraft should be resolved for a given conflict scenario, as shown in Fig. 3.

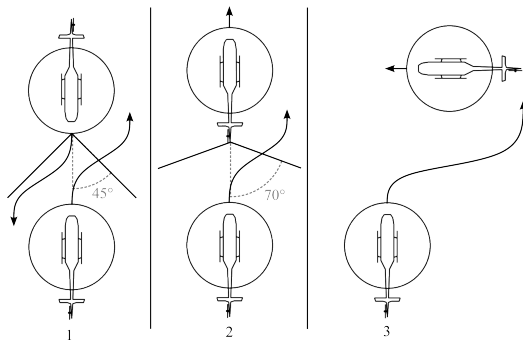


Figure 3: Right-of-Way according to the SERA.3210.

In the scenarios shown above the aircraft with the right-of-way can continue on its path while the other aircraft has to take steps to avoid it. As the aircraft used in this paper is an unmanned and fully automatic UAV it never has the right-of-way and must always take steps to avoid the collision [9]. In cases 1, 2a, 2b and 3a this does not contradict SERA.3210 and gives even more safety margin between the intruder and the UAV as seen in Fig. 4. The orange arrow in 2b shows the path of the UAV according to SERA.3210 and the green arrow shows the path the UAV must take in this scenario.

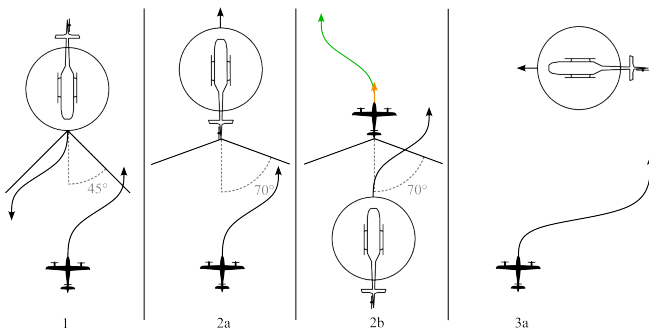


Figure 4: SERA.3210 for UAV, cases 1, 2a, 2b and 3a.

In case 3b and case 3c however the outcome of the conflict scenario depends on the action taken by the intruder. If the

pilot of the intruder aircraft recognizes that the UAV is an UAV and therefore it must take evasive action even though it has right-of-way according to SERA.3210, he will remain on course and not take evasive action himself. In this case the best evasive action for the UAV would be to move to a parallel path to the left, see case 3c in Fig. 5. If however the pilot does not feel safe in this assumption, or does not know that the UAV must always avoid, he might take evasive action according to the SERA.3210 himself. In this case the best avoidance path would be a path to the right, see case 3b in Fig. 5.

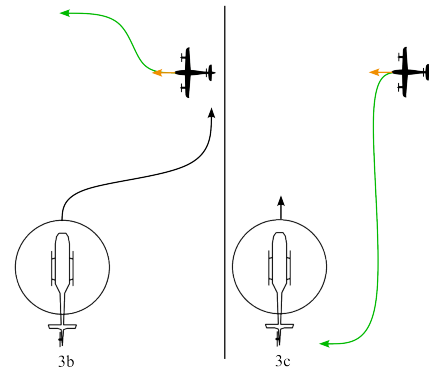


Figure 5: Dilemma in cases 3b and 3c.

As this case has no clear correct solution, other avoidance methods must be used. Otherwise, once the appropriate side for the avoidance maneuver has been determined, a parallel path is planned.

The path is divided into three subpaths: two connecting path segments (1&3) and a parallel path segment (2) as seen in Fig. 6. All parts of the path are planned with the speed v_{UAV} , the UAV is flying at the moment the avoidance path planning is triggered.

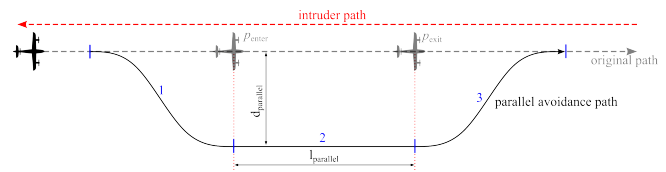


Figure 6: Subpaths of a parallel avoidance path.

The connecting path segments (1&3) are generated using Dubins paths from the current position to the start of the parallel path segment and from the end of the parallel path segment to the rejoin point on the original path [10]. These Dubins paths are planned using the minimum turn radius of the UAV at the given speed. To generate the parallel path segment, the time t_{enter} , position p_{enter} and the heading ψ_{enter} at which the UAV would first enter the intruder's safety zone if it would proceed on the original path

as planned are determined. Next, the time t_{exit} and the position p_{exit} , at which the UAV would leave the intruder's safety zone is calculated. Finally the size of the safety zone of the intruder at t_{exit} , $d_{\text{safety,max}}$ is calculated. Using these parameters and the avoidance side, the parallel path segment is generated by placing a straight path segment of the length $l_{\text{parallel}} = v_{\text{UAV}} \cdot (t_{\text{exit}} - t_{\text{enter}})$ with the heading ψ_{enter} . This straight path segment is then positioned such that its starting point is at the distance $d_{\text{parallel}} = d_{\text{safety,max}}$ to the left or the right of p_{enter} .

The parallel path segment is then combined with the connecting path segment. The second connecting path intersects the original path on a point located one second behind p_{exit} . As p_{exit} is the point from where the original path is collision free, this guarantees a collision free path after rejoining.

The finished parallel avoidance path is checked for validity i.e. is checked for collisions with the intruder as well as other static obstacles and geofence breaches. If the parallel avoidance path is not collision free, due to static obstacles or a second intruder, the distance d_{parallel} is increased by 10% of $d_{\text{safety,max}}$ up to ten times. As this method is not designed for more than one intruder or static obstacles, the generation of the path does not consider them. However, by incrementally increasing the distance of the parallel path segment away from the original path collision free paths are achievable with this method even for such scenarios. Once a generated path is collision free it is chosen as the final avoidance path.

2.3 Avoidance using Speed Variations

The third method used, is the variation of the speed at which the path is flown. This method therefore does not deviate from the original path at all. This poses the advantage, that neither the geofence nor airspaces or ground height have to be taken into consideration for the avoidance maneuver, as these were already taken into account while planning the original path. The disadvantage however is that this method can only solve a very specific set of collision scenarios. Any collision where an intruder's heading is pointing at the UAV i.e. a head-on or a overtaking scenario cannot be solved by varying the speed of the UAV. However, scenarios such as the intersection of the UAV's path by the intruder can be solved very effectively by using this method.

To check whether a speed change can solve the collision, the possible reachable speeds of the UAV starting at the point where the collision avoidance is triggered are calculated in 1m/s increments. This contains decelerations and accelerations. For each of these speeds, starting at the speed closest to the current UAV speed, the movement of the UAV along the path with the new speed as well as the corresponding movement of the intruders is simulated. This is done until either the simulated UAV enters the safety zone of one of the simulated intruders or all simulated intruders start to move away from the simulated UAV. If the simulated UAV enters the safety zone of a simulated intruder, the tested speed is not suitable

to avoid the collision and the next speed must be tested. If all simulated intruders start to move away from the simulated UAV and no collision has occurred until that point, the speed checked is suitable to avoid the collision. The point at which all simulated intruders start to move away from the simulated UAV is then used as the point at which the UAV can start to accelerate or decelerate to the speed originally planned on that part of the path.

2.4 Avoidance using mCOWEX Algorithm

The final collision avoidance method used for multi method local path planner is the mCOWEX algorithm. The detailed explanation of the algorithm can be found in [3]. This algorithm is capable of avoiding collisions with more than one intruder as well as multiple static obstacles. The downside is, that the avoidance path is determined only by the cost function and can therefore differ from avoidance paths that a pilot might take, as seen in Fig. 7. Therefore, the algorithm is only used in scenarios with more than one intruder or if there is an intruder in combination with static obstacles.

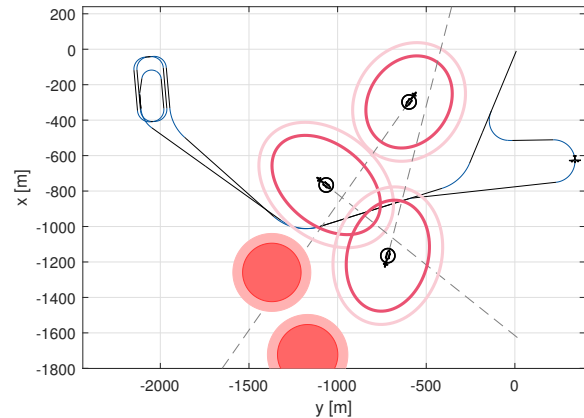


Figure 7: A mCOWEX avoidance path for three intruders and two obstacles, that does comply with SERA.3210, but is collision free.

The mCOWEX algorithm works by expanding circles (denoted as waves from here) into collision free 2D-space within the geofence. This is done by first expanding a wave around the starting point and thereafter expanding new waves on the boundary of an existing wave. Which wave is expanded next is determined by a cost function. The placement of the newly expanding waves, is constrained by the minimal turn radius of the UAV, as the centers of the wave determine the waypoints of the final path. The algorithm stops once the endpoint can be connected to the current wave with a collision free Dubins path. An example of the generation of a collision free path with the mCOWEX algorithm for one intruder is shown in Fig. 8.

If the mCOWEX algorithm does not find a collision free path at the current UAV speed, it is checked to what speed the UAV can decelerate on the original path and still have enough separation from the intruder to complete a 90° turn. A slower speed gives the UAV a smaller minimum turn radius and therefore increases the mCOWEX algorithms chances of success. The mCOWEX algorithm is then restarted with the new starting parameters.

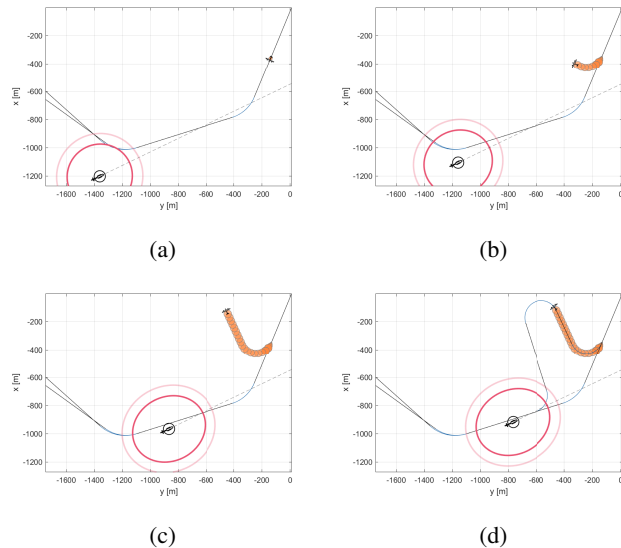


Figure 8: Generation of a path with the mCOWEX algorithm for a head-on collision.

3 AVOIDANCE METHOD SELECTION LOGIC

The logic used to select which of the four methods presented above is used if more than one method provides a collision free and valid path will be explained in this section. Once the local path planner is triggered all methods will simultaneously calculate a path. Once all methods have finished, their path is added to the pool of possible paths, if they have found a valid collision free path. If there is more than one valid path, a criterion has to be used to select which path is best suited for avoiding the collision. The most important criterion when considering possible avoidance paths is to create a successful separation between the UAV and all intruders, obstacles, airspaces, geofences and the ground. As this is already done by the four presented methods a secondary criterion must be found to distinguish between the available paths.

For the multi method local path planner the second criterion is to choose the path that most resembles a human pilots reaction. This criterion is chosen, to simplify the prediction of the future UAV path for human pilots of intruders as well as ATC controllers. If the UAV for example always uses the mCOWEX algorithm first, which finds a path in almost all scenarios, it will theoretically always fly on a safe path. Since this path is defined by a cost function the path may seem com-

pletely erratic to the pilot of the intruder aircraft causing him to fly the intruder plane into a more dangerous situation.

Using this second criterion the paths are selected in the following order: Vertical Avoidance, Parallel Paths, Speed Variation and mCOWEX Algorithm. First, if possible, the path using the vertical avoidance method is chosen. As a pilot of an intruder is easily able to distinguish that the UAV is no longer flying at the same altitude as himself, this method requires the least amount of guesswork by the pilot, if the UAV is indeed on a collision free path. Second, the parallel path method is chosen. This method follows the SERA, so a pilot should also be able to predict the flight path taken by the UAV and act accordingly. Next, the speed variation method is chosen. This does not resemble a conventional avoidance path. It does however follow the original path of the UAV and does not include sudden changes of direction, which may be hard to understand for pilots. If all these paths are not valid, the path generated by the mCOWEX algorithm is used. This path does not consistently comply with SERA, but always provides a safe path, given that all intruders continue flying as they have done at the point the collision avoidance is triggered.

If none of the paths is valid, the same procedures are deployed again using only half of the original safety zone around the intruders. This causes a closer separation but will still guarantee a collision free flight path. If reducing the safety margin does not generate a valid path, the UAV will trigger its flight termination system, which stops all engines and deploys a parachute, to guarantee that no collision occurs.

4 EVALUATION AND DISCUSSION

To evaluate the presented multi method local path planner, first all four path planning methods separately as well as the combination of all four methods will be evaluated by running Monte Carlo simulations with different numbers of intruders and obstacles. The results of these simulations will be used to evaluate the success rates of the different methods for a given avoidance scenario. Thereafter, examples of paths planned for different avoidance scenarios using the multi method path planner will be shown and briefly discussed.

4.1 Success Probability of the Different Path Planning Methods

The Monte Carlo simulations were conducted with four different combinations of zero to two obstacles and one to two intruders. Examples of the scenarios used can be seen in Fig. 9. For the simulation, the starting positions and safety zones¹ of all obstacles and intruders, as well as the intruder's speed and heading were varied. The UAV always flew on the same path, see Fig. 9, with the same starting speed and height profile.

¹The safety zone defined by a safety radius and is shown as a light red for the obstacles and intruders. The intruder's predicted flight path for the next 60 seconds is shown with a dashed gray line.

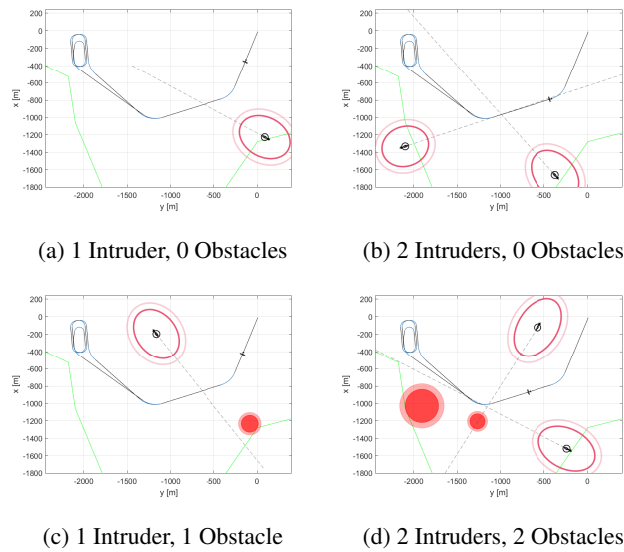


Figure 9: Examples of avoidance scenarios with different number of intruders and obstacles. The original path of the UAV is shown in black for the straight segments and blue for the curved segments.

The success probability of the different methods can be seen in Fig. 10. The figures also include the success probability of the multi method local path planner. As the multi method local path planner only needs one of the methods to succeed, its success probability can be higher, than that of the most successful method. For a given scenario, a successful attempt was either successful with the full safety zone (blue) or using a reduced safety zone (orange).

As Fig. 10 shows, the success rate of the first three path planning methods has a strong dependance on the given scenario.

The vertical method scales poorly with the amount of intruders but is not affected by the number of obstacles. The poor success rate with an increasing number of intruders, is caused by a higher chance of encountering an intruder closer to the current position, which in turn decreases the time the UAV has to climb or descend to avoid the intruder. The number of obstacles does not effect the success rate, as this method does not deviate from the original path, and the original path is free of obstacles.

The method of parallel paths has a higher success rate than the vertical method in all scenarios. This is to be expected, as this method deviates from the original path in the horizontal plane and the UAV is more agile in the horizontal plane in comparison to the vertical plane and changes of speed. Therefore, an avoidance path can be found when intruders are too close to climb or descend to avoid them. This method is designed to follow the SERA.3210, which only describes collision avoidance for one intruder, thus it does not offer a good success rate for scenarios with two intruders. As

the parallel path method does not consider obstacles while generating the path, this method does not scale well with an increasing number of obstacles.

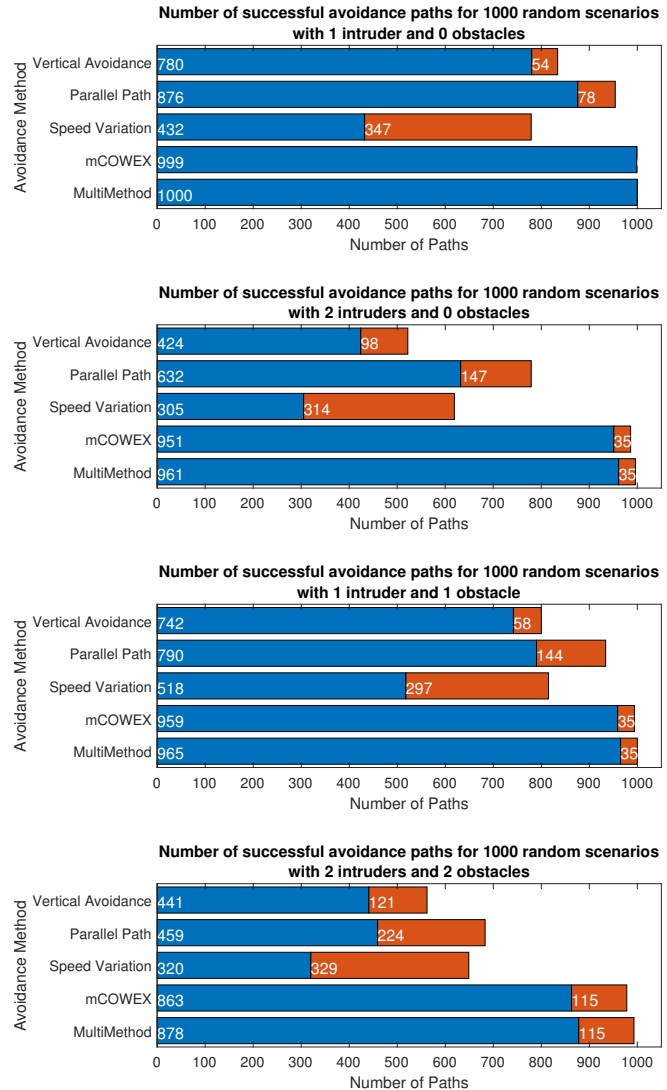


Figure 10: Success rates of all avoidance methods for 1000 randomized avoidance scenarios with different combinations of intruders and obstacles.

The speed variation method scales poorly with the amount of intruders, but is not affected by the amount of obstacles for the same reasons as the vertical method. The greater drop in success rate with the increase from one to two intruders in comparison to the other methods is due to the speed variation method only working for crossing intruders. With the increase from one to two intruders the chances of encountering a head-on or overtaking intruder also double, thus greatly reducing the success rate of the method.

The most successful method is the mCOWEX algorithm, as it takes both intruders and static obstacles into account during path planning. Therefore, its performance only slightly decreases as more obstacles and intruders are added to the scenario. Combining the success rates of all path planning methods to the multi method local path planner, gives a success rate of 99.3% while only needing to reduce the safety margin in 12% of cases. This holds true even in scenarios with two intruders and two obstacles, which are of academic nature, as they are unlikely to occur during real flights. Therefore, the multi method local path planner presents a viable solution to be used as a collision avoidance method for a UAV in automatic BVLOS operation.

4.2 Example Avoidance Scenarios

The following section will give an example avoidance path for each type of the presented avoidance scenarios. First, a vertical avoidance path for a scenario with one intruder and no obstacles is shown in Fig. 11.

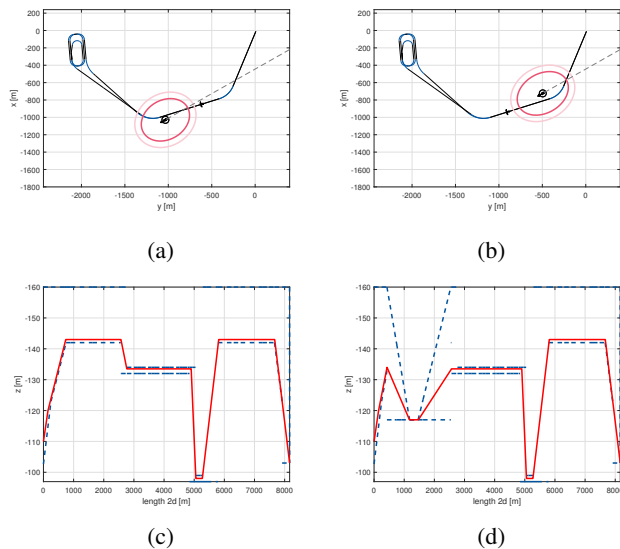


Figure 11: Avoidance Path for a scenario with one intruder and no obstacles using vertical avoidance.

As a vertical avoidance maneuver is difficult to display in 2D, the top view in Fig. 11a and Fig. 11b is expanded by the side view in Fig. 11c and Fig. 11d. The vertical avoidance can be better seen in the side view. The blue lines represent the upper and lower boundaries of the height profile and the red line shows the height at which the UAV flew. In this case the avoidance was accomplished by altering the original height profile, see Fig. 11c, by descending below the intruder between the 2D path lengths of 1200 m and 1500 m, see Fig. 11d. These are the 2D path lengths in which the UAV would have been inside the safety zone of the intruder, if the UAV had followed its original height profile.

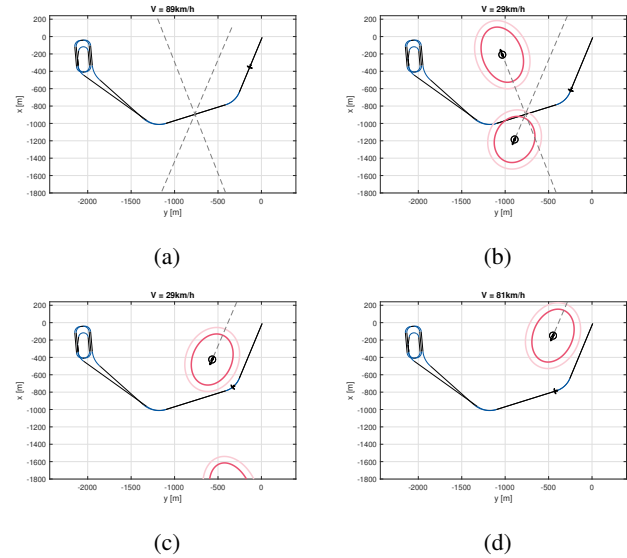


Figure 12: Avoidance Path for a scenario with two intruders and zero obstacles using the speed variation method.

Next, Fig. 12 shows the avoidance of two intruders and no obstacles, using the speed variation method. As seen in the figure, the UAV slows down to avoid both intruders and accelerates to its original speed afterward.

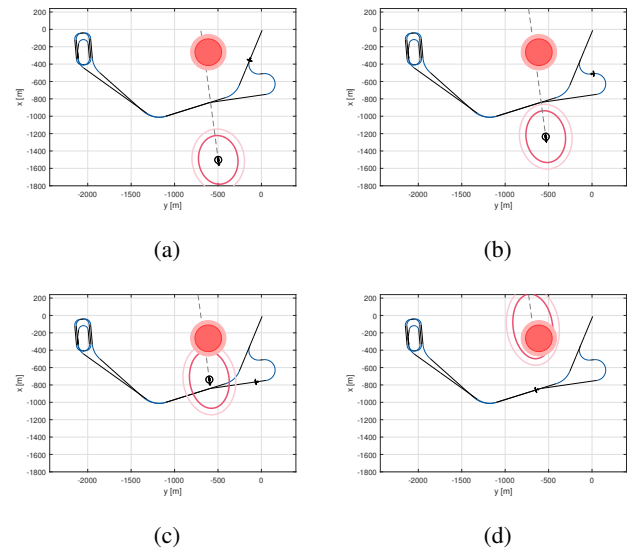


Figure 13: Avoidance Path for a scenario with one intruder and one obstacle using the mCOWEX algorithm.

Figure 13 shows an avoidance path for a scenario with one intruder and one static obstacle using a path generated with the mCOWEX algorithm. The last avoidance path, see Fig. 14, shows an avoidance path generated by the parallel path method. As the obstacles are on to the left of the UAV

and both intruders dictate a parallel avoidance path to the right, the method is able to generate a successful path, even though the scenario is far more complex than what the method was designed to handle.

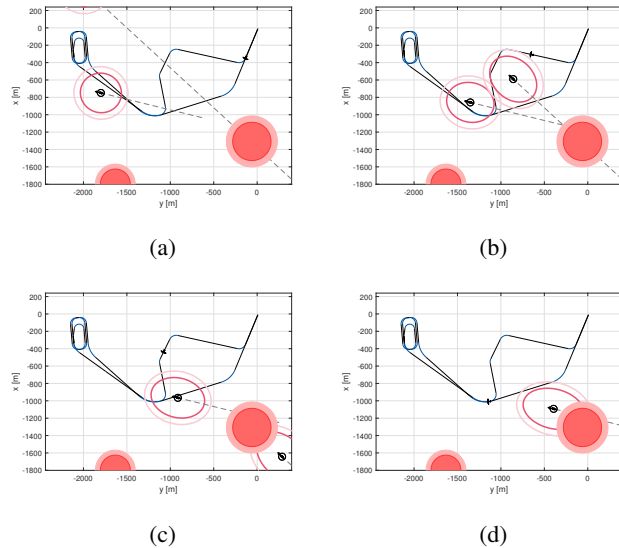


Figure 14: Avoidance Path for a scenario with two intruders and two obstacles using the parallel path method.

In conclusion, the multi method local path planner offers a versatile and capable avoidance method, that can deal with complex scenarios, while also generating efficient and understandable paths in simple avoidance situations.

5 CONCLUSION

This paper presents a multi method local path planner that uses a combination of four different collision avoidance algorithms to generate collision free paths. The four methods used are speed variation on the original path, height profile variation on the original path, avoidance to a parallel side path and the mCOWEX algorithm. First, all methods were presented, and their respective pros and cons were given. Second, the logic that determines, which avoidance path is chosen in case more than one method finds a valid path, was explained. Next, the methods were first evaluated individually by evaluating their success rate in different avoidance scenarios, using Monte Carlo simulations. These same simulations were also used to compare the success rate of the combination of all four methods for the same avoidance scenarios. Finally, examples of avoidance paths generated with all methods of the multi method local path planner were given for different avoidance situations. As the multi method local path planner was able to find collision free paths in 100% of all scenarios it is suitable for use in fully automatic UAV in BVLOS operation.

ACKNOWLEDGEMENTS

Special thanks go to the German Federal Ministry for Digital and Transport (BMDV) for funding and to TÜV Rheinland Consulting GmbH for support and coordination of the EULE project.

REFERENCES

- [1] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.
- [2] I Barz, L Danielmeier, S Seitz, and D Moormann. Kollisionsfreie flugbahnen für kippflügelflugzeuge auf basis des wavefront-algorithmus unter berücksichtigung flugmechanischer beschränkungen. Deutscher Luft-und Raumfahrtkongress, 2020.
- [3] Lennart Danielmeier, Sebastian Seitz, Isabelle Barz, Philipp Hartmann, Max Hartmann, and Dieter Moormann. Modified constrained wavefront expansion path planning algorithm for tilt-wing uav. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 676–685. IEEE, 2022.
- [4] Oussama 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. IEEE, 1985.
- [5] Zhibo He, Chenguang Liu, Xiumin Chu, Rudy R Ne-genborn, and Qing Wu. Dynamic anti-collision a-star algorithm for multi-ship encounter situations. *Applied Ocean Research*, 118:102995, 2022.
- [6] Beobkyyoon Kim, Terry Taewoong Um, Chansu Suh, and Frank C Park. Tangent bundle rrt: A randomized algorithm for constrained motion planning. *Robotica*, 34(1):202–225, 2016.
- [7] R. Marchthaler and S. Dingler. *Kalman-Filter: Einführung in die Zustandsschätzung und ihre Anwendung für eingebettete Systeme*. Springer Fachmedien Wiesbaden, 2017.
- [8] Commission implementing regulation (eu) no 923/2012 of 26 september 2012. *OJ*, L 281:14, 2013-10-13.
- [9] Commission implementing regulation (eu) 2019/947 of 24 may 2019. *OJ*, L 152:45–71, 2019-06-11.
- [10] Lester Eli Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. pages 497–516.

Autonomous Landing, Obstacle Avoidance and Block Recovery for a Quadrotor Drone

Matteo Fei^{*}, Marcos Linares Bayón[†], Andrei Muresan Radu[‡], Wonhee Lee[§], Leandro R. Lustosa
ISAE-SUPAERO, Université de Toulouse, France

ABSTRACT

This technical report presents our proposed solution to the IMAV 2023 Drone Competition. The competition suggests an autonomous block recovery mission using a quadcopter with obstacle-avoidance capabilities. Our drone hardware and software architectures are discussed herein, and, in particular, we introduce a ROS-less custom configuration for intermachine communication between a high-level, low-sampling-rate Companion Computer and a low-level, high-sampling-rate Flight Controller. In addition, the developed controllers and state machines to tackle the different phases of the competition are presented. Finally, the necessary computer vision algorithms for relative navigation amongst obstacles, windows, and blocks are explored.

1 INTRODUCTION

This paper aims to present the drone brought to the IMAV 2023 indoor competition by the IONLAB team. The competition challenge involves collecting and stacking cone blocks with known dimensions and weight. Different path alternatives are to be followed to fetch the blocks, each with its corresponding complexity level and associated score. Each team can decide which path to follow. This paper focuses on the path that involves landing on a rotating platform and passing through a window with a moving obstacle.

This article is organized as follows. Section 2 explains the architecture of the drone hardware while Sec. 3 explores its software counterpart. Section 4 presents the simulations to validate our strategy. Section 5 introduces the state machine of the drone. Section 6 explains the computer vision tasks necessary for relative navigation in a GNSS-denied environment. Finally, Sec. 7 summarises the work presented in the paper and provides findings.

2 HARDWARE ARCHITECTURE

Our platform builds on a pre-built Holybro X500 V2 frame equipped with Pixhawk 6X Autopilot Flight Controller hardware. Except for configuration parameters (i.e., propeller

number and geometry, transmitter mapping, and calibration data for inertial sensors and magnetometer), the corresponding driving PX4 software code has not been modified from the community development tree¹. Instead, an Nvidia Jetson Nano Companion Computer with in-house tailored behavior for guidance, altitude control, and computer vision broadcasts attitude and thrust setpoints to the Pixhawk board. Additionally, stereo and monocular cameras retrieve visual information from the surroundings (see Fig. 13). Finally, a downward-facing LiDAR sensor measures the relative altitude from the ground. Figure 1 summarizes the drone's avionics.

The mechanism chosen to pick the blocks is two 3D-printed pins fixed between the legs of the quadrotor. The pins are attached to a cartesian mechanism that enables them to slide recovering or releasing the cone depending on the section of the competition. The sliding mechanism will be controlled by a servomotor through a circular-to-linear mechanism, being this motor directly connected and controlled to the Pixhawk board.

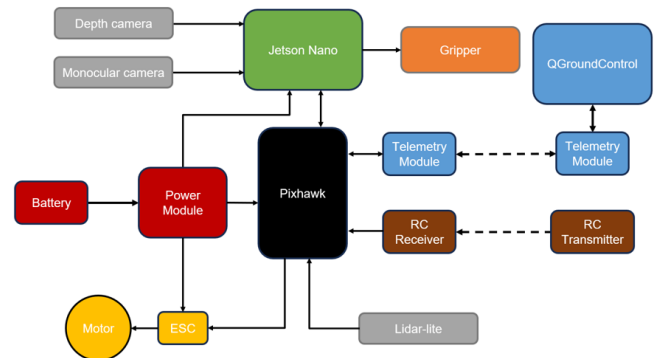


Figure 1: Avionics architecture.

3 SOFTWARE ARCHITECTURE

This section describes the distributed control routines operating asynchronously in the Flight Controller (i.e., Pixhawk) and the Companion Computer (i.e., Jetson Nano) boards. The Flight Controller provides low-level high-frequency (i.e., sampling rate up to 400Hz) sensing, stabilization, and control. At the same time, the Companion Computer provides high-level lower frequency (around 50Hz) guidance routines. The latter also includes computer vision through a

^{*}Corresponding Author: matteo.fei@student.isae-supaero.fr

[†]marcos.linares-bayon@student.isae-supaero.fr

[‡]andrei.muresan-radu@student.isae-supaero.fr

[§]wonhee.lee@student.isae-supaero.fr

¹More specifically, commit hash 3303323971f02cff6eb2fc029562bc4.

dedicated Graphics Processing Unit (GPU) and the state machine that manages mission phases and respective guidance laws switching conditions and mechanisms.

3.1 The Flight Controller Flight Stack

The PX4 software consists of two layers: the flight stack, an estimation and flight control library, and the middleware, a general robotics layer that supports many platforms (e.g., fixed-wing aircraft, rotary-wing copters, rovers), providing communication interfaces and hardware abstractions. The flight control architecture is dependent on the platform. In a quadcopter, for instance, the control flight stack consists of a cascade of Proportional-Integral-Derivative (PID) controllers according to Fig. 2.

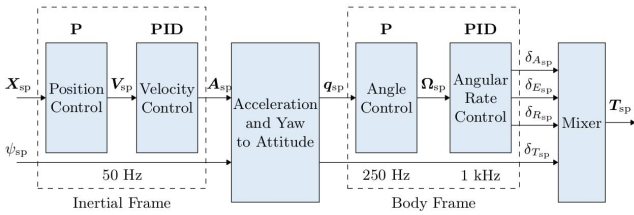


Figure 2: PX4 quadcopter cascaded controller. (Adapted from the official PX4 online documentation [1].)

Each PID controller is a separate process in the Pixhawk 6X board, and all PX4 processes share the same memory space (through NuttX operating system calls). The PID controllers exchange data internally (i.e., in Flight Controller scope) through the uORB messaging middleware. For inter-machine communication, the μ XRCE-DDS middleware exposes Flight Controller uORB topics through a UDP socket interface.

3.2 The Intermachine Communication Bridge

While the commonplace strategy to deploy intermachine communication between the Pixhawk Flight Controller and a Companion Computer is to install the Robot Operating System (ROS) and rely on the native μ XRCE-DDS client on PX4 and respective predeveloped μ XRCE-DDS ROS agent modules on the Companion Computer side, we have decided to develop our own μ XRCE-DDS agent, namely, the **PILION** project, circumventing the need of ROS (see Fig. 3). We avoid ROS usage in our projects due to portability issues: some of our drones operate QNX and other Real-Time operating systems incompatible with ROS. Thankfully, μ XRCE-DDS is compatible with QNX, for instance. Nevertheless, our Companion Computer in this drone does operate Linux Ubuntu.

The Companion Computer broadcasts the desired control setpoints every 20ms to the Flight Controller. In case of intermachine communication failure, a Flight Controller timeout (natively included in PX4) automatically switches to an autonomous hold mode for safety.

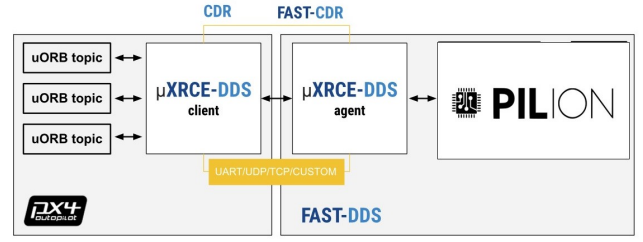


Figure 3: Intermachine interface. (Adapted from the official PX4 documentation [2]).

PX4 provides a set of uORB messages for setting desired setpoints to the drone. The setpoint abstraction level choices go from high-level input, such as desired position, down to desired torque and thrust inputs. According to the chosen setpoint abstraction level, higher-level PID controller modules should be turned off in the Flight Controller to avoid message conflict due to multiple setpoint masters. For example, referring to Fig. 2, if the Companion Computer periodically sends acceleration setpoints, the first two PID controllers of the cascade should be neglected. To give a position, velocity, or acceleration setpoint, PX4 requires a Global Navigation Satellite System (GNSS) receiver. Since the drone participates in an indoor competition, decent GNSS coverage is not guaranteed. Consequently, we chose to have the Companion Computer send attitude setpoints through the uORB message `Vehicle Attitude Setpoint`. This message requires a quaternion setpoint and a normalized collective thrust input. For that to occur, the internal Flight Controller PX4 flight mode should be set to `Offboard`. The PX4 flight mode dictates the PID controller setpoints' origin. In our case, we neglect pilot and ground control setpoints through the offboard mode). Fast switching to stabilized mode (where pilot inputs are taken as velocity setpoints instead) is available for safety through a pilot transmitter switch.

3.3 The Companion Computer Controller

Figure 4 shows a schematic of the overall Companion Computer controller architecture. It achieves position control through a series of PD controllers for each axis. The z -axis controller drives the drone's thrust input, while the x -axis and y -axis controllers drive the pitch and roll angles, respectively. Implementing a yaw controller is unnecessary since this angle is unrelated to any positional input. Finally, roll, pitch, and yaw reference angles are wrapped together as a quaternion setpoint before being sent to the Flight Controller.

Finally, we developed a State Machine in the Companion Computer to adequately tackle the different mission phase requirements. Accordingly, depending on the mission phase, the State Machine decides if relative target position measurements should come from the forward-facing or downward-facing camera computer vision modules.

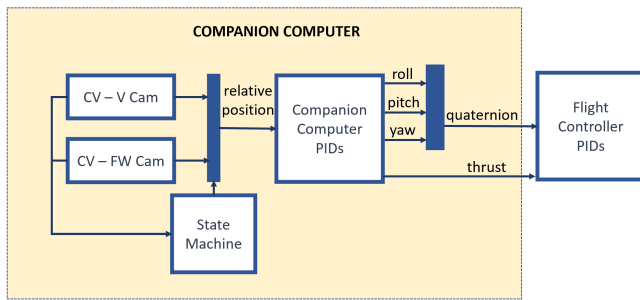


Figure 4: Companion Computer control diagram.

4 SIMULATION

The PX4 development environment provides a Software-In-The-Loop (SITL) simulation environment using the Gazebo Engine to simulate the exact Flight Controller code in a simulated environment with visualization and debugging interfaces available and the possibility of connecting to QGroundControl. The later open-source application provides ground station functionalities. However, a MATLAB model simulating PX4 behavior (drone dynamics plus low-level PID control) was conceived for controller tuning to analyze the Companion Computer PID controllers' response through MATLAB practical functions and toolboxes.

4.1 Tuning Gains for the Companion Computer Controllers

The Companion Controller PID controller gains, which takes a target relative position as input and outputs quaternion and thrust setpoints for the Flight Controller PID controllers, are computed through the pole placement technique. Figure 6 shows the step response outcome on the x -axis.

The dynamical response is satisfactory regarding overshoot, being less than 4%. The performance on the other axes has similar behavior.

4.2 Landing on a Moving Platform

Once the Companion Computer controller is designed, a MATLAB simulation is performed to test its behavior when landing on a moving platform. A state machine manages the landing decision-making process, inspired by others seen in literature [3, 4]. Let the error vector be the difference between the drone's position on the x - y plane – namely, the plane of the platform's rotation – and the platform's position. The drone will track the target (i.e., the platform's position) while keeping a constant altitude. The drone starts descending while following the target when the error norm is below a predefined threshold. If the target is lost, meaning that the error norm becomes higher than the predefined threshold, the drone will return to a given altitude and repeat the process until the landing is fully archived. For this simulation, no computer vision module was implemented, and the platform's position for landing is assumed to be known. Also, the position of the drone itself is considered to be known: in the

competition environment, this is not true since GNSS signal is not available, and instead, the relative difference between the drone and the target is considered. The Simulink model used to archive this simulation is shown in Fig. 5. Finally, the competition platform rotates in a 50 centimeter radius at an angular velocity of $10^\circ/\text{s}$.

Figure 8 shows the simulation results. The drone starts seeing the platform 20 seconds from the start of the simulation and starts to follow the platform. Complete landing is archived in 5 seconds, as shown in Fig. 7. The final error at the moment of landing is 10 centimeters, which is low enough for the competition.

5 STATE MACHINE

We subdivide the proposed complex mission into several tasks that the drone performs in an orderly fashion. This division allows for Companion Computer orderly instructions determination at each point and is implemented as a state machine. Figure 9 illustrates the state machine at the highest level. Starting from the drop zone, the drone flies through a window while avoiding a moving obstacle, then reaches the pickup zone and picks up a cone-shaped load. Then, it returns to the original stack zone while avoiding the same obstacle and places the block on a pile of previous ones or the ground, in the case of a first iteration.

We further break down each one of the blocks of the high-level state machine into programmable commands. Figure 10 depicts the logic for moving through the transit zone. A series of references are established in order of priority for the drone to detect and follow. First, it looks for the window and approaches it. If the window is not found, it goes towards the QR code behind it. If the latter is not detected, it follows the line on the ground. If the line is not found, it flies toward the known magnetic trajectory of the target while still searching for any of the previous visual references. When the window is detected and approached, it stops at a 1 m distance, where the forward camera can see the whole screen. It waits in front of it until the moving obstacle is cleared, and it advances following the guideline. While crossing the window, altitude is no longer read with the Lidar range sensor but with the barometer to avoid noise caused by sudden attitude changes due to the window's frame. After passing the window, a similar cascade of references approaches the QR code at the end of the transit lane.

Figure 11 shows the algorithm followed to pick up the block. Once the forward camera sees the ArUco code, the drone approaches it until it is visible to the vertical down-facing camera. At that point, the drone starts following the circular motion of the platform until it is consistently over the code, a point at which it starts descending. The drone then lands on the platform, the cone is mounted manually, and the drone takes off again to a predefined mission altitude. In case of loss of QR code visuals, the drone returns to mission altitude and repeats the process.

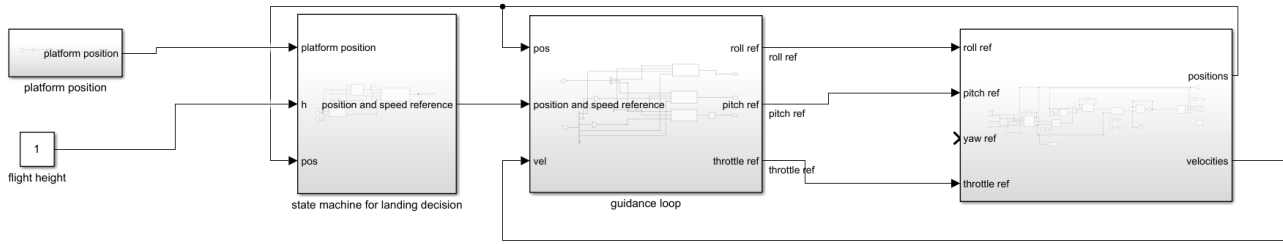


Figure 5: The Simulink model of the Companion Computer control diagram. On the left side, the state machine and the guidance laws represent the Companion Computer controller, while the block on the right represents Flight Controller controllers and the drone dynamics.

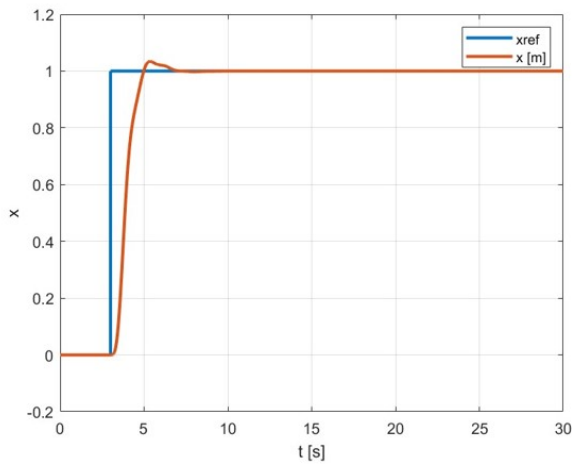


Figure 6: Step response on x -axis.

The last of the high-level blocks of the state machine is the release of the cone on top of the cone pile. At the end of the transit back, the drone searches for the QR code of the drop zone, reaches the position on top of it, and slides slightly left. For the first block, it simply descends to an altitude calculated based on the cone height and releases it. Then, it takes off again at the right altitude and yaw and proceeds towards fetching the next cone. For the consecutive ones, the drone approaches the QR code, looks for the stack, and centers itself on top using the vertical camera. The drone descends to a height calculated from the number of cones stacked and releases the load. Safety protocols are implemented in case of loss of target visuals or centering.

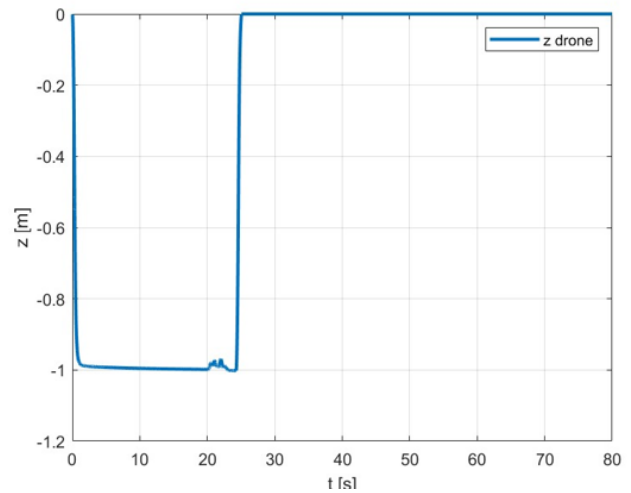


Figure 7: z -axis plot.

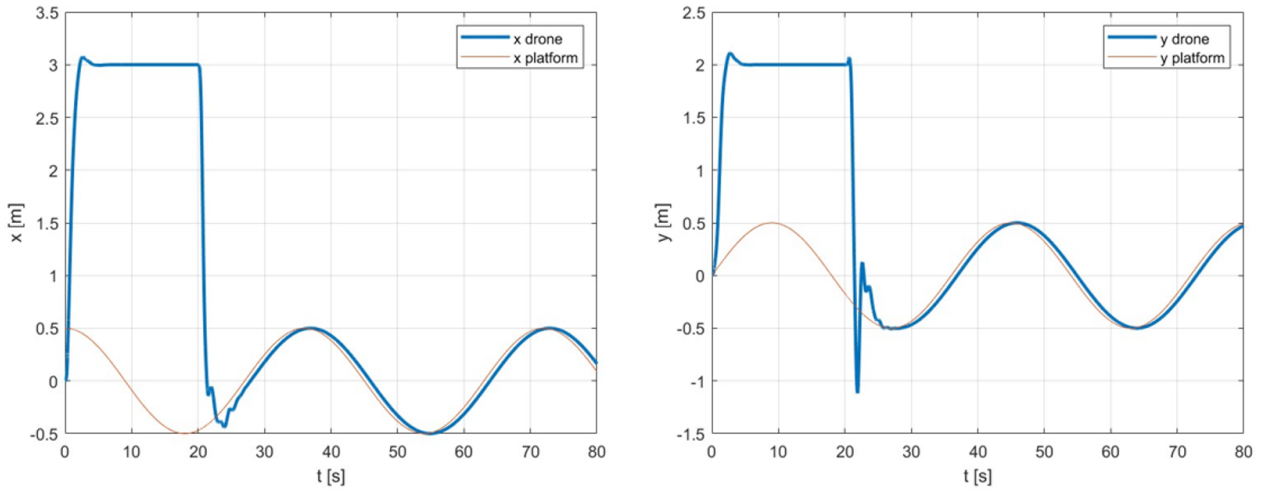
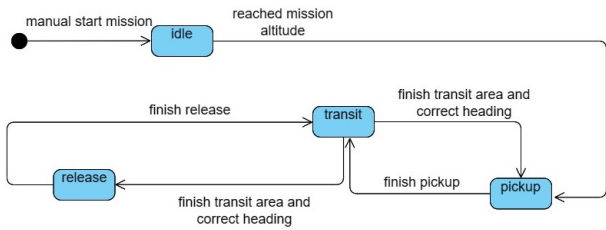
Figure 8: x -axis and y -axis plot.

Figure 9: High level state machine.

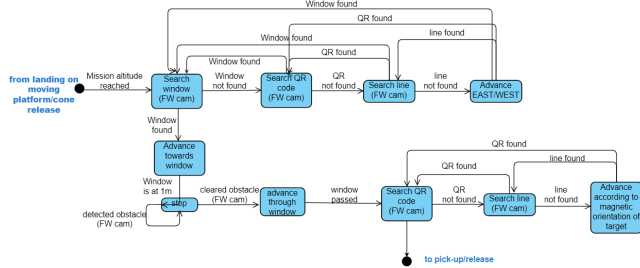


Figure 10: Transit state machine.

6 COMPUTER VISION

The purpose of the computer vision is to provide a setpoint for an active control loop, based on the visual information acquired by cameras. The setpoint is a 3D relative pose given by the special Euclidean group (SE(3)) or its component.

6.1 Architecture

The drone uses two cameras; one forward-looking Intel Realsense D455 depth camera, and one down-looking fisheye monocular camera. The field of views (FoV) of the camera

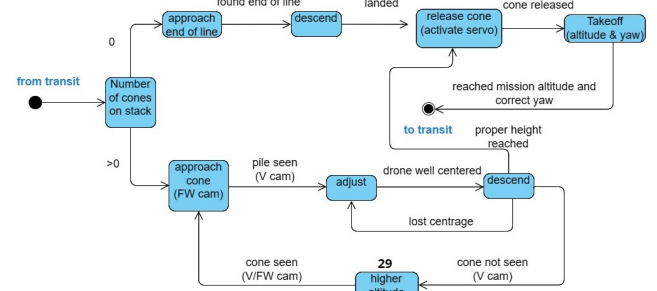


Figure 11: Block collection state machine.

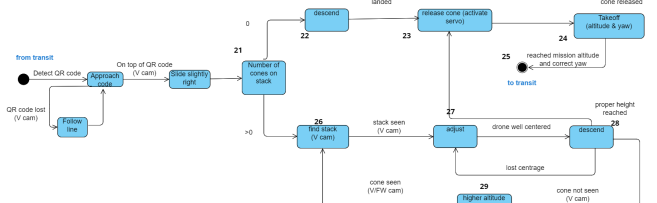


Figure 12: Block release state machine.

when looked from the side are depicted in Fig. 13.

The depth camera allows to acquire the depth data of the scene directly without using computationally costly algorithms. The fisheye camera reduces the blind spot between the two cameras.

The computer vision software module comprises of five main classes. The name and output of each class is provided in Table 1.

To allow various camera types, each class is independent of the camera model. Different camera models are dealt by

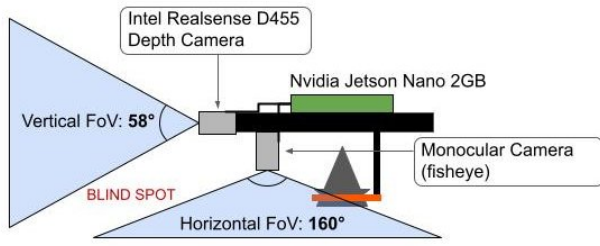


Figure 13: Drone architecture from computer vision perspective and side field of views (FoV) of cameras.

Table 1: Computer vision module classes with their outputs.

Class	Output
ArUco_Detector	Transform matrix (SE3) from marker to camera (T_{cm})
Window_Detector	Transform matrix (SE3) from window to camera (T_{cw})
Cone_Detector	3D position of cone in camera coordinate system.
Release_Decider	Boolean which is true when it is decided that the cone is safe to be released.
Line_Tracer	Approximate relative bearing of the line with respect to the drone.

the abstract camera representation as used by some visual odometry (VO) or simultaneous localization and mapping (SLAM) algorithms such as ORB-SLAM3 [5]. Considering the FoVs of the camera used, the depth camera is modelled by the Brown[6]-Conrady[7] model, and the fisheye monocular camera by the Kannala-Brandt [8] model.

Figure 14 shows the essential state machine with active classes for each mission phase.

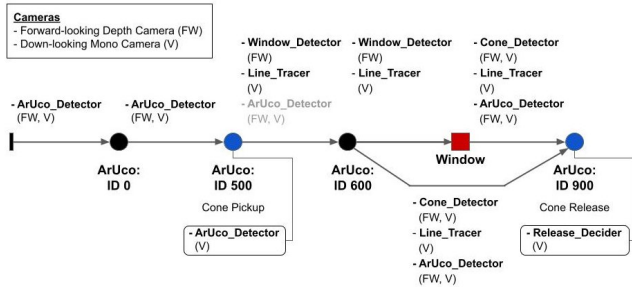


Figure 14: High-level state machine from computer vision perspective.

6.2 Marker Detection and Pose Estimation

The goal of the marker detection is to provide the drone with the relative 6 DoF pose of the drone with respect to the marker. The perspective-n-point (PnP) algorithm estimates the pose of the camera, given n number of 3D points of an object in the camera coordinate system and corresponding 2D pixel points in the image.

Since a marker is planar, a special type of PnP algorithms called infinitesimal plane-based pose estimation [9] (IPPE) can be used. In this case, minimum 4 pairs of points are needed, and many open-source libraries such as OpenCV al-

lows to acquire positions of 4 vertices of a marker as well as its ID. IPPE can yield at most two possible solutions.

Along with the solution ambiguity of IPPE, other factors may affect the pose estimation quality, such as the distance from the marker to camera with respect to its focal length[10]. To improve the robustness on estimation, three enhancement methods were applied:

1) initial pose guess is obtained by solving PnP with random sample consensus[11] (RANSAC) to remove outliers.

2) if the difference between the guess obtained from 1) and the previous pose is larger than a threshold, the guess is ignored.

3) if the pose difference is less than the threshold, the guess obtained from 1) is iteratively refined using Levenberg-Marquardt[12][13] method.

Figure 15 shows the visual result of the ArUco marker pose estimation.

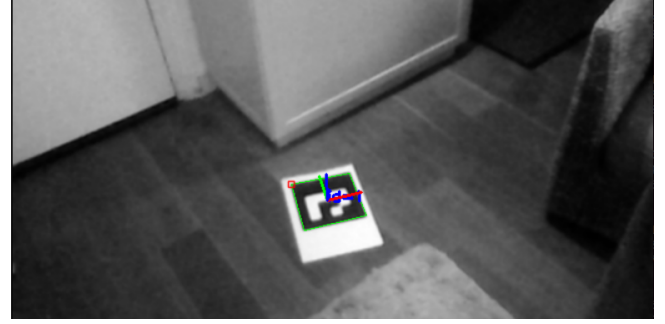


Figure 15: Detected ArUco marker and its estimated pose represented by the axes.

6.3 Cone Detection, Tracking and Pose Estimation

The goal of cone detection is to provide the relative 3D position of the target cone which the drone aims to approach. Two approaches were considered to detect the cone: 1) traditional method based on the image processing using geometric characteristics and extracted features, 2) deep learning method based on the neural network trained with custom dataset. Both approaches have their own weaknesses. The former is more prone to change in the scene such as perspective, and the latter is more dependent on the dataset quality.

The mean average precision (mAP) is a metric to evaluate the performance of an object detection algorithm. The mAP of deep learning algorithms surpass most traditional algorithms [14]. Therefore, You Look Only Once (YOLO) object detector was selected as the basis for the cone detection. Specifically, YOLOv4-tiny was chosen to achieve faster inference under limited computing resources.

In addition to detection, tracking is required to follow only one cone out of multiple cones, if they exist. Hence, after the detection of cones, one cone is locked for tracking based on a certain criteria, such as the distance from the image center. Then, a (Channel and Spatial Reliability Tracking) CSRT

[15] tracker is used to track the one. When tracking is lost, a new cone is locked. Figure 16 shows the cone detection and tracking result.

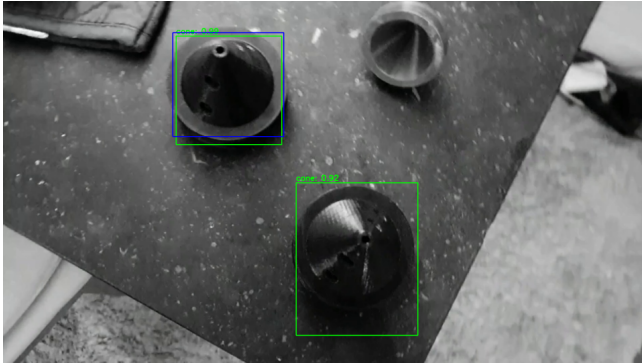


Figure 16: Detected cones (Green) using YOLOv4-tiny trained with custom cone image dataset, and the locked cone (Blue) for tracking using CSRT tracker.

During the tracking, the relative position of the cone can be estimated by unprojection. For the depth camera, the needed depth is directly obtained. For a monocular camera, the known dimensions of the cone can be used to infer the depth.

6.4 Cone Release

The goal of cone release decision is to signal the drone when it is believed to be safe to release the cone. First, the desired 3D position of the cone is projected to the image. The bounding box of the projected cone will be named as the target box. The bounding box of the cone output by the tracker will be named as the tracker box. Then, the cone is released when the intersection over union (IoU) between the target box and the tracker box is over a certain threshold. Figure 17 shows the cone before and after the release decision.

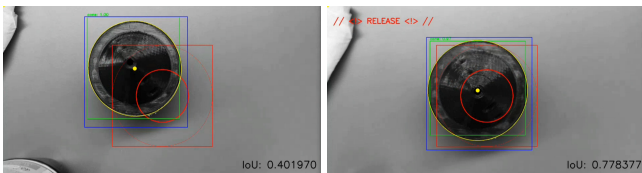


Figure 17: Cone release decision based on the IoU between the target and tracking bounding boxes. Before release (left), after release (right).

6.5 Window Detection and Pose Estimation

Window detection can be classified as an object detection problem in computer vision. Since the exact shape of the window is not provided, the image processing approach was chosen over the deep learning one.

The depth image obtained by the depth camera is first thresholded by the depth boundaries from 0.3 to 4 m to

get the foreground. This image is binarized with Canny edge detection [16], then quadrilaterals are found using the Ramer–Douglas–Peucker algorithm [17, 18]. Figure 18 shows the scene with a window, and the detected window.



Figure 18: Scene with a window (left), and detected window using depth camera (right).

However, this method has following expected limitations.

- 1) if the window is not perfectly quadrilateral, the possibility of detection is scarce.
- 2) the frequent noise in the depth image near the window leads to 1).
- 3) depending on the threshold used for the Canny edge detection and on the lighting condition, some edges might not be detected. This case also leads to 1).

To overcome these limitations, a more robust algorithm is required.

Once the contour of the window is determined, the relative pose between the window and the camera can be found using the PnP algorithm. If the orientation is not needed, the relative translation can be found by unprojecting the centroid of the contour.

6.6 Colored Line Tracing

The goal of line detection is to provide the relative bearing of the line with respect to the drone. In addition to the bearing, the approximate relative 2D position of the line also needs to be provided to keep the drone lined up.

To extract only the blobs of a specific color from an image, thresholding based on color can be used. However, color is heavily dependent on the lighting condition. Therefore, to improve the robustness [19], red-green-blue (RGB) color space is first converted to hue-saturation-value (HSV) color space. Still, the line can be detected as fragmented blobs due to the varying lighting condition. To reduce this fragmentation, morphological transformations are applied. Next, contours having reasonable areas are sought, along with smallest rectangles that enclose them. The rectangle that has the maximum fineness ratio in the image is assumed as the guiding rectangle. Finally, the relative bearing is set to the angle between the longer side of the rectangle and the vertical axis of the image. The geometric center of the guiding rectangle is used to estimate the relative 2D position of the line with respect to the camera. Figure 19 shows the detected red line and the center and relative bearing derived from the guiding rectangle.

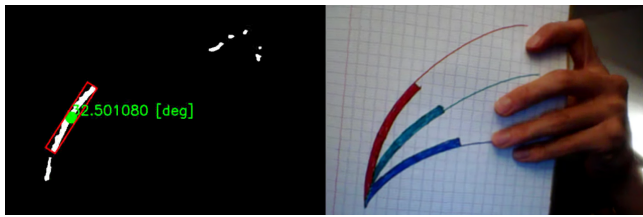


Figure 19: Detected red line and its computed relative bearing (left) in the image of hand-drawn lines of three different colors (right).

7 CONCLUSION

This draft paper intends to give some insight into the work process that went into preparing the drone for the IMAV 2023 competition. In it several aspects of the project are explained, from the architecture, to the dynamics involved, to the avionics that goes with it. Given the degree of autonomy required for the drone, computer vision is also a fundamental part of the project. Additionally, an oversight of the logical process represented by the state machine is also explained, corresponding to the actions performed in the different phases of the mission. It is worth noting that the paper is not a complete walk-through of the work done since as of the day of the delivery, the project is not finished.

REFERENCES

- [1] https://docs.px4.io/main/en/flight_stack/controller_diagrams.html#fixed-wing-attitude-controller.
- [2] https://docs.px4.io/main/en/ros/ros2_comm.html.
- [3] Alvika Gautam, Mandeep Singh, Pedda Baliyarasimhuni Sujit, and Srikanth Saripalli. Autonomous quadcopter landing on a moving target. *Sensors*, 22(3), 2022.
- [4] Pengyu Wang, Chaoqun Wang, Jiankun Wang, and Max Q.-H. Meng. Quadrotor autonomous landing on moving platform. *Procedia Computer Science*, 209:40–49, 2022. Proceedings of the 2022 International Symposium on Biomimetic Intelligence and Robotics (ISBIR).
- [5] Carlos Campos, Richard Elvira, Juan J. Gomez Rodriguez, Jose M. M. Montiel, and Juan D. Tardos. ORB-SLAM3: An accurate open-source library for visual, visual-inertial, and multimap SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890, dec 2021.
- [6] Dean Brown. Decentering distortion of lenses. 1966.
- [7] A. E. Conrady. Decentred Lens-Systems. *Monthly Notices of the Royal Astronomical Society*, 79(5):384–390, 03 1919.
- [8] J. Kannala and S.S. Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1335–1340, 2006.
- [9] Toby Collins and Adrien Bartoli. Infinitesimal plane-based pose estimation. *International Journal of Computer Vision*, 109(3):252–286, 2014.
- [10] Denis Oberkampf, Daniel DeMenthon, and Larry Davis. Iterative pose estimation using coplanar feature points. *Computer Vision and Image Understanding*, 63:495–511, 05 1996.
- [11] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.
- [12] Kenneth Levenberg. A method for the solution of certain non – linear problems in least squares. *Quarterly of Applied Mathematics*, 2:164–168, 1944.
- [13] Donald W. Marquardt, E. I. duPont, Bennett R, and George C Burrell. Marquardt d w. an algorithm for least-squares estimation of nonlinear parameters. *j. soc. indust. appl. math.* 11:431-41, 1963. 2004.
- [14] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey, 2023.
- [15] Alan Lukežič, Tomáš Vojř, Luka Čehovin Zajc, Jiří Matas, and Matej Kristan. Discriminative correlation filter tracker with channel and spatial reliability. *International Journal of Computer Vision*, 126(7):671–688, jan 2018.
- [16] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.
- [17] Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244–256, 1972.
- [18] David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.
- [19] Aisha Ajmal, Christopher Hollitt, Marcus Frean, and Harith Al-Sahaf. A comparison of rgb and hsv colour spaces for visual attention models. *2018 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pages 1–6, 2018.

Automating Fixed Wing Forced Landings with Offline Reinforcement Learning

*Alexander Quessy, Thomas Richardson, and Sebastian East
University of Bristol, Queens Building, United Kingdom, BS8 1TR

ABSTRACT

Executing off-field landings in unprepared locations is a crucial skill for single-engine piston fixed-wing aircrew, particularly during sudden engine failures. The unpredictability and challenging nature of such failures often leads to flight crew overload and accidents. Further, as autonomous air vehicle capabilities advance, incorporating the ability to autonomously land following engine failure is likely to become a vital design requirement.

This paper presents a unified forced landing methodology that leverages reinforcement learning (RL) to develop adaptable policies for a wide range of forced landing scenarios, with a focus on model-based offline RL. The research outlines a graphical simulation environment, procedures, and algorithms for training an RL-based controller. The effectiveness of the proposed approach is demonstrated, highlighting that offline RL is a promising solution for designing controllers capable of executing glide approaches into predetermined locations.

1 INTRODUCTION

Executing off-field landings in unprepared locations is a key skill required by all Single Engine Piston (SEP) Fixed Wing (FW) aircrew as engine failures or fires can occur suddenly and unexpectedly at any point in flight. Executing a dead-stick/no-power landing is challenging and requires the aircrew to fly the aircraft within an appropriate airspeed range to avoid stalling or over-speeding. They must also navigate to a safe landing spot, all while not being able to maintain a fixed altitude for a pre-planned approach. The suddenness of engine failures can complicate matters further, as the aircrew must quickly identify a landing site to navigate to before losing too much altitude and the option to land. The suitability of a landing site may only become apparent when the aircraft is low and near the final landing location. At this point, there may not be enough energy available to land elsewhere.

Regulations reflect this limitation requiring operators to glide-clear [1], remaining 1,000 feet above built-up-areas and preventing scheduled commercial air transport on SEP FW

aircraft, within Europe. Despite forced landing training being an integral part of SEP aircrew training and assessment, incidents involving engine failures still tragically result in fatal accidents due to a loss of control and failure to identify suitable landing sites [2, 3, 4].

The numerous concurrent tasks that must be completed to safely execute a forced landing place large cognitive demands upon the aircrew, these include, but are not limited to:

- Identifying that a loss of power has occurred and taking appropriate immediate actions to prevent loss of control.
- Control of the aircraft through the use of visual reference and instruments, which may become inoperative if reliant on the engine driven vacuum pump.
- Visual identification of an appropriate landing location, which often depends on the aircraft's attitude, requiring the aircrew to maneuver the aircraft to visualize the locations depending upon the airframe geometry.
- Navigating towards the designated landing zone whilst accounting for wind direction and speed.
- Attempting engine restart, shutdown or fire control/suppression procedures as appropriate.
- Coordinating with air traffic to arrange search and rescue operations after the forced landing.

As a result, a common cause of engine failure incidents leading to accidents is aircrew overload, where startle following an engine failure leads to either a loss-of-control or a failure to position the aircraft to make an approach towards an appropriate landing location [5]. Further, as the size and capabilities of autonomous pilot-less aircraft gradually increase, SEP FW aircraft are a sensible platform owing to their ubiquity and low operational cost (compared to multi-engine and turbine types). It is reasonable to expect that autonomous off-field engine-out landings will become a certification requirement to enable these types of operations.

Designing methods to automatically solve this task is challenging as it requires a balance between online exploration and exploitation, while acting upon limited information with high-stakes outcomes. Reinforcement learning (RL) offers a solution to this problem by enabling policies to be learned that can generalize to complex tasks through the use of previous experiences.

*Email address: aq15777@bristol.ac.uk

This work presents a unified forced landing methodology that aims to serve two purposes: assisting aircrew in executing forced landings to help reduce startle, surprise and overload and establishing the groundwork for autonomous fixed-wing forced landings. Our contributions can be summarized as follows:

- Section 3 presents the general challenge of landing a fixed-wing aircraft in an off-field location, and describes a 2D graphical simulation environment to simulate aircraft landing following an engine failure in various locations. Source code for this environment at the following URL . . .
- Section 4 details the procedures and algorithms used to generate offline data for training an RL-based controller, their relevance to the fixed-wing landing problem, and their application to model-based offline RL.
- Section 5 showcases the effectiveness of offline RL in designing controllers capable of executing glide approaches into predetermined locations by learning from demonstrations. The performance of the learned controller is evaluated in a variety of both in-distribution and out-of-distribution scenarios.

2 RELATED WORK

2.1 Fixed-Wing Forced Landings

Previous research on fixed-wing off-field landings is fairly limited. [6], focused on identifying landing sites through edge detection algorithms applied to real-world data. The authors of [6] used 3D Dubin's path-planning methods [7] to generate trajectories, whereas our work employs sampling-based approaches, constraining the aircraft to a fixed rate of descent and airspeed. Unlike our approach, [6] allows the aircraft's pitch angle, and therefore rate of descent, to vary. Additionally, [8] and [9] considered landing site reachability analysis.

2.2 Model Based Offline Reinforcement Learning

RL is a type of machine learning that uses a reward signal from an agent to train a policy capable of achieving task specific goals [10]. Combining deep neural network based function approximators with RL has proven to be hugely successful to solve a variety of complex tasks including games [11], robotics [12] and natural language processing [13]. This work focuses on the offline RL setting, which involves learning from a dataset of offline demonstrations. Behavior Cloning (BC) is the simplest version of this, where the policy is directly learned from (s, a) state-action pairs, under the assumption that the demonstrations come from an optimal policy demonstrator π^* [14].

Unlike model-free RL methods such as SAC [15] and DDPG [16], model-based offline RL methods, such as MoREL [17] and MOPO [18], learn both a model and a

policy, enabling online planning similar to Model Predictive Control (MPC) [19]. This approach results in an improved online policy compared to pure BC examples, thanks to the added benefits of planning. This work uses Model-Based Offline Planning (MBOP) [20] to learn a value function that enables zero-shot online adaption to new goals or constraints. MBOP also provides improved interpretability by allowing the loss and accuracy of the dynamics and value function to be viewed directly throughout the training process.

3 PROBLEM STATEMENT

This section provides a detailed description of the motivation and methodology behind the FW forced landing simulation environment. It also explains how this problem can be formulated as a Markov Decision Process (MDP) and how RL can be used to solve it. The source code for the environment is available at the following URL [released with full paper] for use by other researchers.

3.1 Fixed-Wing Environment

The *flyer* simulation environment models aircraft dynamics using a 3D Dubin's car, represented by the dynamic equations specified in equation 1. The aircraft's position is represented by a 3D point in space, denoted as (x, y, z) , with a heading angle specified as θ . The aircraft's velocity, V , rate of descent $\dot{Z}$, and the simulation time step δt are pre-defined and fixed. Wind and atmospheric effects are not incorporated in the simulation, hence all headings are considered as tracks and airspeeds as ground speeds.

Only bank angle ϕ is used to control the aircraft's position and heading, constrained to a maximum of 30 degrees. The aircraft's turn-rate is then calculated using c_ϕ which causes the aircraft to turn at a standard rate of $3^\circ/s$ at 15° angle of bank. Bank angle is then used as the action in the MDP formulation.

$$\begin{aligned} x_{t+1} &= x_t + V \cos(\theta) \delta t \\ y_{t+1} &= y_t + V \sin(\theta) \delta t \\ z_{t+1} &= z_t - \dot{Z} \delta t \\ \theta_{t+1} &= \theta_t + c_\phi \phi \delta t \end{aligned} \quad (1)$$

Observations consist of either an 800x800 RGB image captured from a downward-facing camera with a 30-degree field of view, gimbal to be level along the xy earth plane from the aircraft, as shown in Figure 1. Additionally, the aircraft's state is represented as a tuple (x, y, z, θ) . For this report, only the aircraft's position and attitude observation is considered, but the RGB image is left within the simulation and provided source code for future work.

The termination criterion is met when the aircraft comes to a stop on the ground, which can be either due to a crashed state or when the ground roll is completed. A crashed state occurs when the aircraft's preassigned *health* property falls to zero, by default this is an integer set to 1. Colliding with an obstacle like a rock usually causes damage reducing health

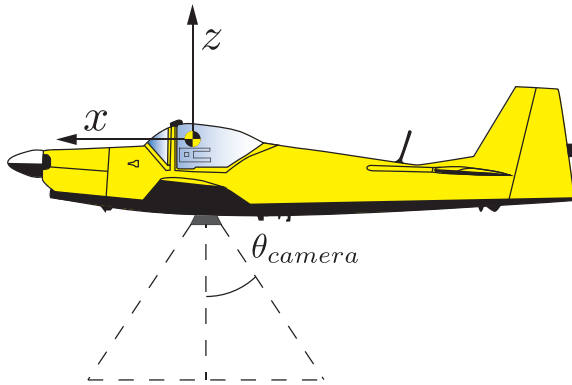


Figure 1: Image observations are generated by a simulated camera, $\theta_{camera} = 30^\circ$.

by one. Some obstacles like crops are destroyed in this event, others like trees persist inflicting damage at the next time step.

The simulated world comprises several landing terrain types each with varying levels of complexity, from easiest to hardest:

1. *Grass*: an open grass field, the entire area is unobstructed and safe for landing.
2. *Orchard*: a grass field containing sparse uniformly distributed apple trees, collision with a tree causes the aircraft to crash. The field also contains stumps and rocks that cause one damage as the aircraft passes over the object.
3. *Crops*: a dirt field containing crops, collision with a crop deals one damage and removes the crop. Crops are clustered together as a 2D Gaussian distribution.
4. *Forest*: a dirt field containing trees, collision with a tree causes the aircraft to crash. Trees are clustered together as a 2D Gaussian distribution.
5. *Beach*: a narrow sand area containing banana trees. The area is thin and only exists where water meets land.
6. *Water*: a water area, any landing in this region results in a crash.

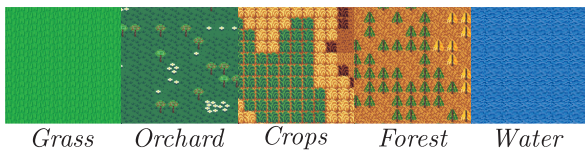


Figure 2: Landing terrain types in the flyer environment, with order of landing difficulty/danger increasing from left to right

The Flyer environment is procedurally generated offering the user large amounts of control as to how the environment is

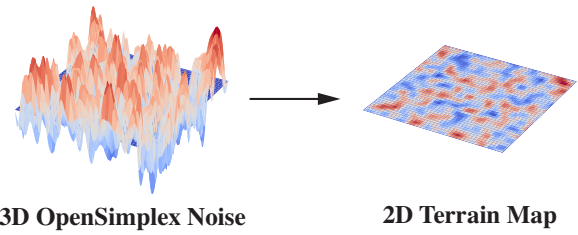


Figure 3: OpenSimplex (Perlin) noise used to generate land and water tile types on the map

constructed, whilst also allowing for a huge range of environmental diversity. 3D OpenSimplex Noise, similar to Perlin noise [21], is initially used to smoothly separate the environment into land and sea and beach terrain types with beach being placed along the intersection. The noise function returns a single value for each (x, y, z) point in 3D space, sweeping across all $(x, y, z = 0)$ positions and returning the noise value at each. If the noise value is less than the water noise threshold the tile is assigned as water, otherwise it is a land type of tile, this is shown in Figure 2.

To generate fields $n_{partition}$ points are randomly placed across the ground xy plane and a nearest neighbor Voronoi decomposition [22] is used to create regions. Each region is assigned a terrain type using a multi-noulli distribution based upon the prior probability of each terrain type existing within the environment, a $(256, 256)$ texture map is created as a result. Objects, such as trees and bushes, are placed within terrain types using the probability distribution for each object within the terrain type.

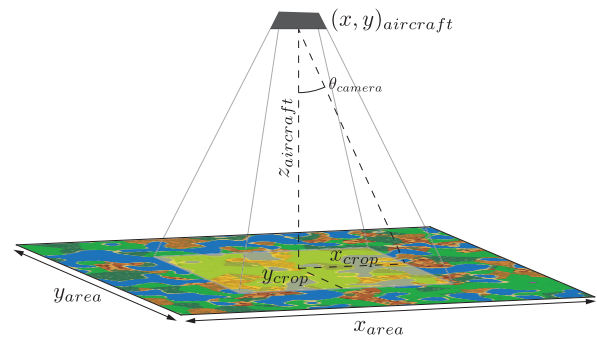


Figure 4: Crop to render aircraft image observation

A base image is rendered by tiling 16×16 RGB textures onto the $(256, 256)$ texture map, objects are placed on-top of this image by selectively replacing pixels in appropriate locations. The resulting 4096×4096 base image is cropped based on the aircraft camera's location. This image size is set using $z \tan(\theta_{camera})$ and the crop centered around the aircraft's location, as shown in Figure 4. The 30° camera angle used for the aircraft crop simulation is equivalent to a

35mm full frame camera lens.

3.2 MDP Formulation

Based on our simulation we can define the problem of fixed wing landings as an MDP, defined by a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma, \mu)$. The goal is to learn a policy $\pi(\mathbf{a}|\mathbf{s})$ that maximizes the expected cumulative discounted reward of the MDP.

- $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces respectively.
- $P(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ maps states and actions to a probability distribution over subsequent states.
- $R(\mathbf{s}, \mathbf{a})$ represents the reward function, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$.
- $\gamma \in (0, 1]$ is the discount factor.
- $\pi_\beta(\mathbf{a}|\mathbf{s})$ is the behavior policy used to generate the dataset $\mathcal{D} \sim P_{\pi_\beta}$, which is itself used to train the online policy $\pi_\phi(\mathbf{a}|\mathbf{s})$.
- $s_0 \sim \mu$ is the starting distribution of the MDP.

The objective of RL is to find the policy π^* that maximizes the expected discounted sum of rewards:

$$\pi^*(s_t) = \arg \max_{a \in \mathcal{A}} \left\{ \sum_{t=0}^{\infty} \gamma^t r(s_t, \pi^*(s_t)) \right\}$$

In the `Flyer` MDP reward is sparse and binary:

$$R(s) = \begin{cases} 1, & \text{if safe-landing} \\ 0, & \text{if crashed} \end{cases}$$

Assuming that the real environment is not available during training, we have access only to a dataset of demonstrations $\mathcal{D}$ containing tuples of (s_t, a_t, r_t, s_{t+1}) . The objective in offline learning is to learn a policy that can achieve zero-shot online task adaption by using the offline dataset. It is also assumed that the policy used to generate these demonstrations, while effective in reaching the goal, is suboptimal.

4 METHODOLOGY

This section outlines the methods used to locate a suitable landing site and train an RL-based controller to identify and execute an approach into the destination. The process is viewed as a 2-step approach that involves generating an offline dataset from simulation to train the controller, followed by training an offline RL controller to execute the approach into the destination.

To generate a dataset of sub-optimal landing trajectories, we use an Importance Sampling (IS), Monte-Carlo Tree Search (MCTS) based algorithm where the value of each node is found using the L_2 distance from a target position and optimal trajectory, found by backtracking along nodes connected by edges with the maximal value. The target position is found by searching for the Pole of Inaccessibility (PoI) [23] on all grass arrays using the grass terrain map.

4.1 Target Position Search

Algorithm 1 provides a description of the procedure used to calculate the PoI. We assume access to the full terrain map creating a binary array, where positions with grass fields are assigned 1 and everything else is 0. A point is assigned to be the PoI if the sub-array formed by expanding linearly around the point being tested of distance $dist$ is all grass with value 1, and the sub-array does not exceed the bounds of the problem space. We show this search procedure graphically in figure 5.

Algorithm 1 PoI Search

Require: T_{grass} grass terrain map

```

1:  $dist_{max} \leftarrow 0$ 
2: for  $(i_x, i_y)$  in  $T$  do
3:    $dist = dist_{max} + 1$ 
4:   while BOUNDSCHECK( $i_x, i_y, t, dist$ ) do
5:     if  $T_{grass}[i_x : i_x + dist, i_y : i_y + dist] == 1$  then
6:        $dist_{max} = dist$ 
7:        $\mathbf{x}_{PoI} = [i_x, i_y, 0]$ 
8:     end if
9:   end while
10: end for
11: return  $\mathbf{x}_{PoI}$ 
```

Algorithm 2 Within Bounds Check

```

1: procedure BOUNDSCHECK( $i_x, i_y, array, dist$ )
2:   if  $i_x - dist < 0$  then
3:     return False
4:   end if
5:   if  $i_y - dist < 0$  then
6:     return False
7:   end if
8:   if  $i_x + dist > array.xlength$  then
9:     return False
10:  end if
11:  if  $i_y + dist > array.ylength$  then
12:    return False
13:  end if
14: end procedure
```

4.2 Monte-Carlo Tree-Search

An IS-MCTS algorithm [24] is used to generate a trajectory from the initial starting position $\mathbf{x}_{start}$ to the final goal position $\mathbf{x}_{goal}$. The action sampling in this algorithm is weighted based on the distance from the goal state. Figure 6 illustrates a simplified example of the IS-MCTS algorithm. After initialization IS-MCTS is a 3-step process of *selecting* which $n_{samples}$ edges to expand along the horizon length H_n . The flyer environment is used to *simulate* the roll-out of each selected node by H_n steps. The tree is then *expanded* by adding the new edges and vertices into the tree and the

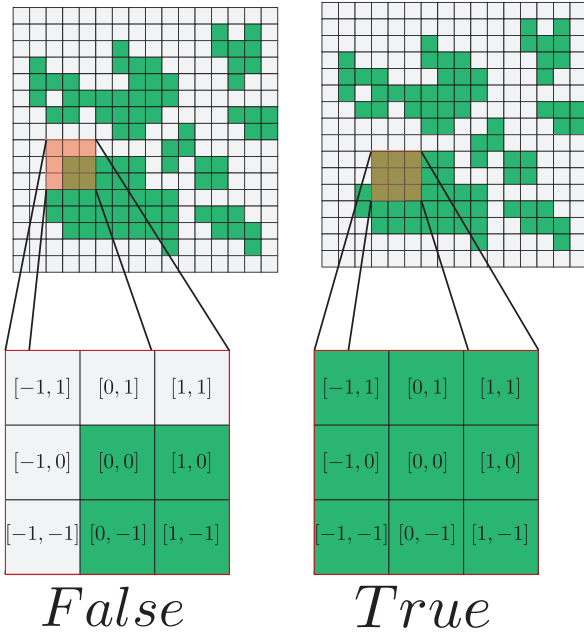


Figure 5: Target Point Grass Check

probability of each node is updated to reflect this weighting. Once the maximum number of steps has been sampled, back-propagation is performed along nodes with the highest value to calculate the optimal trajectory.

Algorithm 3 provides the full IS-MCTS algorithm used in the flyer environment where $\mathbf{x}_{goal} = \mathbf{x}_{Pol}$, found using algorithm 1. The following functions are used through IS-MCTS to obtain useful demonstrations:

- f_{dyn} : Aircraft dynamics from the flyer environment, as described in equation 1.
- f_u : Control sampling function, modelled as a Gaussian distribution where the mean is the last bank angle and variance is fixed at 0.2, $u_t \sim \mathcal{N}(\mu = u_{t-1}, \sigma = 0.2)$.
- f_{constr} : Space constraint function that returns *true* if (x, y, z) position values are in the range $[[0 < x < 255], [0 < y < 255], [0 < z < 64]]$ and *false* otherwise.
- f_x : A multinomial distribution, where the probability mass function can be formulated using the gamma function as:

$$\frac{\Gamma \sum_i x_i + 1}{\prod_i \Gamma(x_i + 1)} \prod_{i=1}^k p_i^{x_i}$$

The vertex in V with the maximum value is closest to the goal state $\mathbf{x}_{goal}$. From there, the edges contained in G are backtracked to select the optimal trajectory τ_{opt} . This is illustrated graphically in Figure 7, sampled with hyperparameters: $N_{steps} = 2000$, $N_{samples} = 100$, $H_n = 10$.

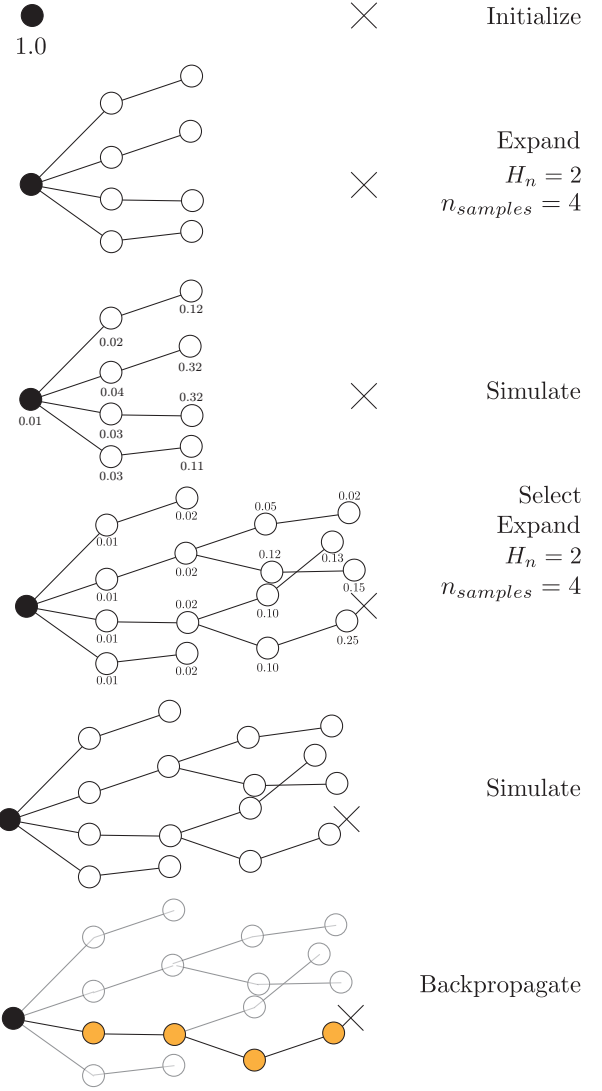


Figure 6: Simplified 2D IS-MCTS algorithm

4.3 Model Based Offline Planning

Using the data generated by the previously described goal searching and navigation algorithms, MBOP (Model Based Offline Planning) a model-based offline RL algorithm that combines a BC policy with an iterative guided n -step shooting method. Classical supervised learning is used to train the model on the dataset $\mathcal{D}$ containing $n_{episodes}$, this includes 3 components:

1. A one-step dynamics model, $f_{dyn} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S} \times \mathbb{R}$, such that $(r_t, s_t) = f_{dyn}(s_t, a_t)$.
2. A behavior cloned policy, $\pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$, such that $a_t = \pi(s_t, a_{t-1})$.
3. A truncated value function, $V : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, providing the expected return over horizon N of taking action a in state s , such that $\hat{R}_H = V(s_t, a_{t-1})$.

Algorithm 3 IS-MCTS

Require: $\mathbf{x}_{start}$, $\mathbf{x}_{goal}$, H_n , $N_{samples}$, N_{steps} , f_{dyn} , $f_u(\mathbf{x}) = \Pr(\mathbf{U} = \mathbf{u} | \mathbf{X} = \mathbf{x})$, f_{constr}
 $G = (\mathbf{x}_{start}, \emptyset)$ $\triangleright$ Create list of edges and vertices
 $V = \{\mathbf{x}_{start} : \|\mathbf{x}_{goal} - \mathbf{x}_{start}\|_2\}$ $\triangleright$ Create dictionary of vertex values
 $p_v \leftarrow \{\mathbf{x}_{start} : \frac{V[\mathbf{x}_{start}]}{\sum V}\}$ $\triangleright$ Probability of selecting an edge v at point $\mathbf{x}$
for N_{steps} **do**
 for i in N_{values} **do**
 $p_v[\mathbf{x}_i] = \frac{V[\mathbf{x}_i]}{\sum V}$ $\triangleright$ Update probability
 end for
 $f_{\mathbf{x}} = \Pr(\mathbf{x}_0, \dots, \mathbf{x}_k; p_v[\mathbf{x}_0], \dots, p_v[\mathbf{x}_k])$
 for i in $N_{samples}$ **do**
 $\mathbf{x}_i = f_{\mathbf{x}}()$ $\triangleright$ Sample from multinomial distribution
 for i_h in H_n **do**
 $\mathbf{u}_{i+i_h} \leftarrow f_{sample}(\mathbf{x}_{i+i_h}, \mathbf{u}_{i+i_h})$
 $\mathbf{x}_{i+i_h+1} \leftarrow f_{dyn}(\mathbf{u}_{i+i_h} | \mathbf{x}_{i+i_h})$
 if $f_{constr}(\mathbf{x}_{i+i_h+1}) == False$ **then**
 $G \cup (\mathbf{x}_{i+i_h}, \{\mathbf{x}_{i+i_h} : \mathbf{x}_{i+i_h+1}\})$
 $V \cup \{\mathbf{x}_{i+i_h+1} : \|\mathbf{x}_{goal} - \mathbf{x}_{i+i_h+1}\|_2\}$
 end if
 end for
 end for
end for

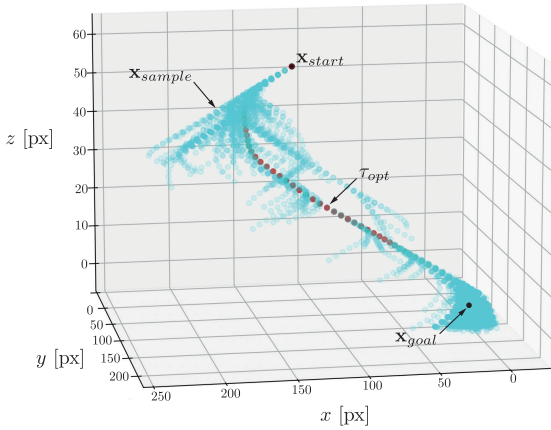


Figure 7: MCTS Results: sampled points are shown in blue and the final trajectory points in red. Vertices not shown for clarity.

Each component is represented with a deep neural network, with input and output layers of the same size as the function shown above, and 2 hidden layers of size 500. After experimentation, we found that ensemble network representations were not required in comparison to the original MBOP algorithm [20]. The samples from MCTS are obtained from a uniformly distributed cube with a start posi-

tion $\mathbf{x}_{start} = [x, y, z]$, where $x \sim \mathcal{U}[x_0 - 10, x_0 + 10]$, $y \sim \mathcal{U}[y_0 - 10, y_0 + 10]$, $z \sim \mathcal{U}[z_0, z_0 - 10]$, and $\mathbf{x}_0 = [128, 128, 64]$, which is the center-point of the map. The target position for all examples is $\mathbf{x}_{goal} = [14, 198, 0]$. Only the MCTS samples that successfully land within the target field are used to train MBOP. The 3 components described above are trained over 1×10^6 steps using 500 episodes, as shown in Figure 8.

Online MBOP-TrajOpt is used to sample trajectories over a horizon length H_n using a combination of sample based Model Predictive Control (MPC) and the behavior cloned policy π . Mixing of the noisy behavior cloned action $a_t = \pi(s_t, a_{t-1}) + \epsilon$, and the best last action $\tau_{a_i=\min(t, H-1)}$ is controlled with a mixing parameter β . Algorithm 4 provides a full description of the online MBOP trajectory optimization procedure.

Algorithm 4 MBOP-TrajOpt

Require: Dynamics Model f_d , Policy π , Value V , Horizon H_n , MDP Dynamics P , MDP termination T
 $\mathbf{s} \sim \mu$ $\triangleright$ Initial state
 $\tau_a = [0, \dots, 0_{H_n}]$
 $terminated = False$
while $terminated == False$ **do**
 $\mathbf{R}_N = 0_N$ $\triangleright N$ trajectory returns
 $\mathbf{A}_{N,H} = 0_{N,H}$ $\triangleright N$ action trajectories of length H
 for $n = 1 \dots N$ **do**
 $\mathbf{s}_1 = \mathbf{s}, a_0 = \tau_a, R = 0$
 for $t = 1 \dots H_n$ **do**
 $\epsilon \sim \mathcal{N}(0, \sigma^2)$
 $a_t = \pi(s_t, a_{t-1}) + \epsilon$ $\triangleright$ BC Action Sampling
 $\mathbf{A}_{n,t} = (1 - \beta)a_t + \beta\tau_{a_i=\min(t, H-1)}$
 $(s_{t+1}, r_{t+1}) = f_{dyn}(s_t, \mathbf{A}_{n,t})$
 $R = R + r_{t+1}$
 end for
 $\mathbf{R}_n = R + V(s_{H_n+1}, \mathbf{A}_{n,H})$
 end for
 $\tau_a = \frac{\sum_{n=1}^N e^{\kappa \mathbf{R}_n} \mathbf{A}_{n,t+1}}{\sum_{n=1}^N e^{\kappa \mathbf{R}_n}}$
 $\mathbf{s} = P(\mathbf{s}, \tau_{a_0})$
 $terminated = T(\mathbf{s})$
end while

5 EXPERIMENTS

Results comparing MCTS to pure BC and MBOP with TrajOpt are presented in Figure 9. There is clear improvement in both the reliability and accuracy of the task with MBOP and BC compared to MCTS. It was found that a 100% task success rate could not be achieved using BC or MBOP due to constraint violations resulting from crashing into obstacles or leaving the boundaries of the flyer environment, as defined in f_{constr} .

1. Operating too far from the known dynamics so plans

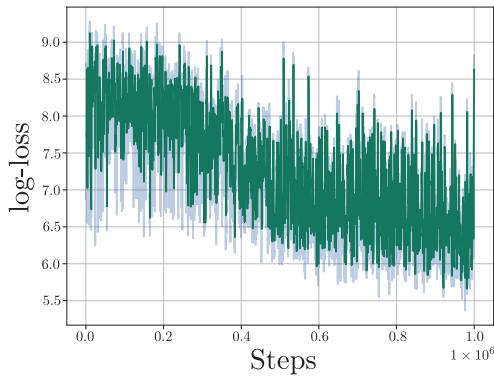
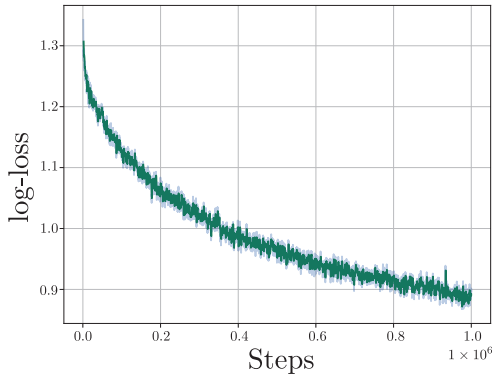
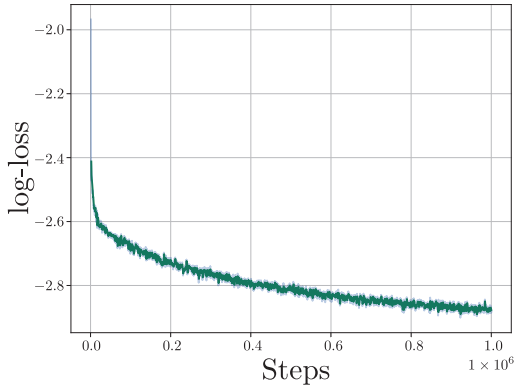
(a) Value V Loss(b) Policy π Loss(c) Dynamics f_{dyn} Loss

Figure 8: Loss Plots for each of MBOP's components, 100 data point rolling average is shown in green and the underlying data in blue.

are not accurate.

2. We violated the constraints of the flyer environment (as described in $f_{constraint}$).

NOTE TO REVIEWER: Further results will be presented in final paper including training over a larger number of environment seeds, along with a comparison of the effect of

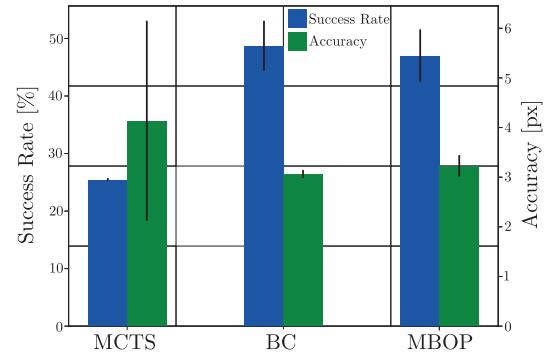


Figure 9: Comparison of methods over 5 seeds, using 100 samples, 1SD is shown with error bars.

dataset quantity on MBOPs accuracy and final task performance. We will also include the full hyper-parameters in a table within the appendix. We will also include the URL link to a public github repository

6 CONCLUSIONS AND FUTURE WORK

This work presents a methodology for autonomous landing control based entirely on offline data. Although the datasets used were obtained from computer simulations, future research would benefit from using real-world offline data collected from an actual aircraft. The study demonstrates how the planning function enables detection of possible constraint violations by the aircraft, making our approach useful as an assistive tool for flight crew.

REFERENCES

- [1] J Hanafin. (uk) standardised european rules of the air, exceptions to the minimum height requirements. *ORS4*, 01, 2021.
- [2] UK Air Accidents Investigation Branch. Aaib investigation to grumman aa-5, g-bbsa. *Air Accidents Investigation Branch*, 2022. Loss of power after takeoff, Teesside International Airport, 25 September 2021.
- [3] UK Air Accidents Investigation Branch. Aaib investigation to piper pa-46-350p (modified), g-hyza. *Air Accidents Investigation Branch*, 2022. Propulsion system failure and forced landing, one mile north of Cranfield Airport, 29 April 2021.
- [4] NTSB. Report era14la383, n50xv. *National Transportation Safety Board*, 2014.
- [5] Javier Rivera, Andrew B. Talone, Claas Tido Boesser, Florian Jentsch, and Michelle Yeh. Startle and surprise on the flight deck: Similarities, differences, and prevalence. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 58(1):1047–1051, 2014.

- [6] Luis Mejias, Daniel Fitzgerald, Pillar Eng, and Xi Liu. Forced landing technologies for unmanned aerial vehicles : towards safer operations. 2009.
- [7] Giuseppe Ambrosino, Marco Ariola, Umberto Ciniglio, Federico Corrado, Alfredo Pironti, and Micol A. De Virgilio. Algorithms for 3d uav path generation and tracking. *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 5275–5280, 2006.
- [8] Matthew Coombes, Wen-Hua Chen, and Peter Render. Reachability analysis of landing sites for forced landing of a uas in wind using trochoidal turn paths. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 62–71, 2015.
- [9] Matthew Coombes, Wen-Hua Chen, and Peter Render. Landing site reachability in a forced landing of unmanned aircraft in wind. 10 2016.
- [10] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.
- [11] David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.
- [12] Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in challenging terrains using egocentric vision, 2022.
- [13] OpenAI. Gpt-4 technical report, 2023.
- [14] Dean A. Pomerleau. *ALVINN: An Autonomous Land Vehicle in a Neural Network*, page 305–313. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989.
- [15] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018.
- [16] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2016.
- [17] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel : Model-based offline reinforcement learning. *CoRR*, abs/2005.05951, 2020.
- [18] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: model-based offline policy optimization. *CoRR*, abs/2005.13239, 2020.
- [19] Manfred Morari, Carlos E. Garcia, and David M. Prett. Model predictive control: Theory and practice. *IFAC Proceedings Volumes*, 21(4):1–12, 1988. IFAC Workshop on Model Based Process Control, Atlanta, GA, USA, 13-14 June.
- [20] Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning. *CoRR*, abs/2008.05556, 2020.
- [21] Ken Perlin. An image synthesizer. *SIGGRAPH Comput. Graph.*, 19(3):287–296, jul 1985.
- [22] Anton Francois. *Voronoi diagrams of semi-algebraic sets*. PhD thesis, University of British Columbia, 2004.
- [23] Daniel Garcia-Castellanos and Umberto Lombardo. Poles of inaccessibility: A calculation algorithm for the remotest places on earth. *Scottish Geographical Journal*, 123(3):227–233, 2007.
- [24] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition, 2007.

6.1 Referencing to literature

APPENDIX A: DATA

If appendices are necessary, they appear at the end of the document. Use ‘apsection’ instead of ‘section’.

APPENDIX B: MORE DATA

Even more data.

Trajectory Optimization for Fully Actuated Hexacopters: Enhancing Maneuverability and Applications

Mohamad Hachem^{*1}, Thierry Miquel^{†1}, Murat Bronz^{‡1}, and Clément Roos^{§2}

¹ENAC, Université de Toulouse, Toulouse, France

²ONERA, The French Aerospace Lab, Toulouse, France

ABSTRACT

Trajectory optimization is a challenging task in the robotics community. Several factors need to be taken into account when generating a feasible optimized trajectory. The optimization process heavily relies on the dynamic model of the system. Currently, there are various drone designs available, categorized based on their actuation status. In this study, we apply a trajectory optimization technique to a fully actuated hexacopter (FA-Hex), which is a new application to the best of our knowledge. This type of vehicle has been successfully integrated into several practical applications. Unlike the under-actuated hexacopter (UA-Hex), the FA-Hex can perform maneuvers with minimal banking angles, significantly enhancing the drone's maneuverability. Our research focuses specifically on trajectory optimization for the FA-Hex and demonstrates the adaptability of our method to different scenarios. We discuss two specific applications: a drone filming without a gimbal joint and a drone with a cable-suspended pendulum. We compare the simulation results with the UA-Hex model to highlight the differences in maneuverability between the two systems. The trajectory optimization is performed offline using CasADi in the MATLAB framework.

1 INTRODUCTION

Aerial robotic systems have found extensive applications in various fields, including package delivery, photography, rescue operations, and construction inspection [1], [2]. These missions have led to the development of different types of vehicles tailored to meet specific requirements. Nowadays, aerial systems utilize cable-suspended loads for package delivery in various environments. Additionally, vehicles de-

signed for photography and inspection purposes often integrate a gimbal joint to ensure optimal camera positioning. In these cases, minimizing drone weight and optimizing trajectory and energy consumption are essential objectives. However, striking a balance between control and vehicle design is critical to achieving optimal performance.

Aerial vehicle systems can be classified based on their actuation level as under-actuated (UA), fully-actuated (FA), and over-actuated (OV) systems [3]. A quad-rotor drone serves as a common example of an UA system, which relies on total thrust and three torques as virtual control inputs. However, these virtual control inputs are insufficient to directly control all 6 degrees of freedom (DoF) of the system. On the other hand, a FA system possesses a number of virtual controllers equal to the DoF, matching the number of actuators. Conversely, OV drones have an excess number of actuators, enabling them to handle actuator failures using various control allocation techniques. FA vehicles have a more extensive range of maneuvers compared to UA ones. Some drones can adapt to online actuation changes by employing additional servo motors that consider the tilt angle of the motors. Indeed, hexacopters can be classified as either UA or FA systems based on their actuator configuration. Hexacopters with actuators in one plane, which typically means they have fixed propellers, are considered UA systems. On the other hand, hexacopters equipped with tilting actuators, often known as coaxial hexacopters or hexacopters with tilting propellers, are recognized as FA systems. The ability of the latter to tilt their actuators provides them with greater control authority and maneuverability, allowing them to be fully actuated and perform more complex maneuvers compared to the former.

This work's primary contribution lies in its flexible approach to solving trajectory optimization problems, which can be adapted to suit a wide range of drone applications. The emphasis lies on scenarios where drones track specific objects through photography. In such cases, it is vital for the drone to maintain a clear field of view of the target by moving with minimal banking angles. Improving the trajectory sent to the low-level controller enhances the drone's ability to accomplish assigned tasks effectively. Additionally, the FA-Hex can be used for package delivery with the help of a cable-suspended pendulum mechanism. This setup enables

^{*}mohamad.hachem@enac.fr

[†]thierry.miquel@enac.fr

[‡]murat.bronz@enac.fr

[§]clement.roos@onera.fr

the drone to reduce oscillations and perform precise maneuvers, even in challenging and static environments.

1.1 Design and Control Strategies

Various vehicle design and control strategies have been developed and utilized to manage the dynamics of FA drones. Employing these vehicles offers the key advantage of independent attitude and altitude control by decoupling their dynamics [4].

Non-linear control based on full geometric control, ensuring precise position tracking, is applied by [5] and [6]. Stability of the resulting controller is established using Lyapunov techniques. Furthermore, [7] proposes an optimal design for the tilted angle based on the active set of possible wrenches exerted by the FA drone, using a feedback linearization control approach with a PID controller to decouple position and orientation control. Another approach is presented in [8], by adding a tilting mechanism to the motors of the UA drone, incorporating two servo actuators to rotate the rotor's angle in radial and tangential directions around the arm. Feedback-linearization control is employed for managing the drone's dynamics. To handle environmental perturbations, [9], uses adaptive control for the FA-Hex drone. Similarly, [10] proposes adaptive sliding mode control for FA-Hex drones with a cable-suspended pendulum, compensating for the inability to measure the load's position by developing an extended gain observer-based adaptive sliding mode approach.

1.2 Trajectory Optimization

Generating a reference trajectory to be tracked by the lower-level controllers is a demanding task when working with FA vehicles. Currently, the literature review on trajectory optimization primarily focuses on UA vehicles. Therefore, it becomes a challenging problem to develop an optimal controller that generates desired positions and orientations for FA vehicles. The research in this area can be categorized into two main approaches. Firstly, a differential flatness approach is utilized to obtain a smooth and feasible trajectory [11, 12, 13, 14]. Secondly, the dynamical model is incorporated as a constraint in the optimization problem [15, 16, 17]. Utilizing the differential flatness approach in formulating the optimization problem has shown to reduce computational time. It has been demonstrated by [14] that the UA quad-rotor drones are differential flat systems, which can be characterized by the following flat output vector: position (x, y, z) and heading (ψ) , of the system. In contrast, FA vehicles are considered to be differential flat based on the full dynamics of the drone. In our approach, we have chosen to incorporate the dynamical system as a constraint in the optimization problem. Differential flatness is utilized to generate the trajectory of an UA vehicle, which is then used as an initial condition for the optimization problem of the FA-Hex vehicle. This overview is the basis of our contribution to the study of a trajectory optimization problem for FA vehicles, taking into account specific objectives.

We organize the remainder of the paper as follows. In Sections 2.1 and 2.2, we present the mathematical model of the FA-Hex and the extended model that incorporates a cable-suspended pendulum, and in Section 3, we formulate the proposed trajectory optimization problem. In Section 4 the simulation results are presented based on the defined optimization problem.

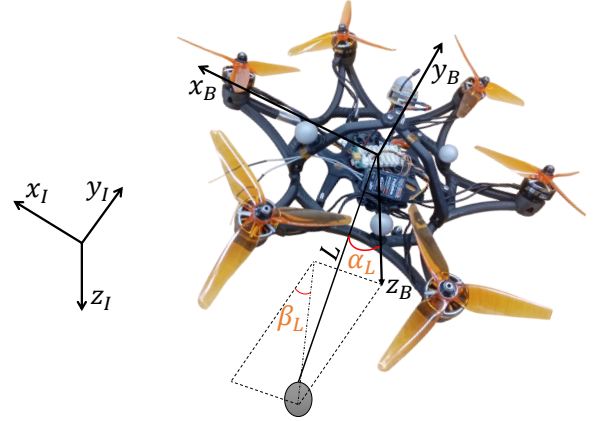


Figure 1: Fully Actuated Hexacopter attached to a payload with Inertial ($\mathcal{F}_I$) and Body ($\mathcal{F}_B$) Frames.

2 MODEL AND SYSTEM VARIABLES

The importance of the dynamical system in achieving workable results, as mentioned earlier, makes it necessary to define the mathematical model of the system we're looking at. In this section, we present the dynamic model of the FA-Hex and its extension to include the cable-suspended pendulum. The model is derived from Newton's Euler equation, while taking into account the following assumptions:

- The drone is assumed to be a rigid body.
- The drone is assumed to be symmetric with respect to its axis, and its inertia matrix is diagonal and denoted

$$\text{as } I_B = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}.$$

2.1 FA-Hex Modeling

The modeling approach of the drone involves two main frames: the inertial frame ($\mathcal{F}_I$) defined by (x_I, y_I, z_I) axis, and the body frame ($\mathcal{F}_B$) defined by (x_B, y_B, z_B) axis, as illustrated in Figure 1. The rotation between $\mathcal{F}_I$ and $\mathcal{F}_B$ is defined by the Euler angles $\mu = [\phi, \theta, \psi]^T$ and the corresponding rotational matrix $R(\mu)$ where:

$$R(\mu) = \begin{bmatrix} C_\psi C_\theta & C_\psi S_\theta S_\phi - S_\psi C_\phi & C_\psi S_\theta C_\phi + S_\psi S_\phi \\ S_\psi C_\theta & S_\psi S_\theta S_\phi + C_\psi C_\phi & S_\psi S_\theta C_\phi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix}$$

with the notations $C_* = \cos(*)$ and $S_* = \sin(*)$.

The position of the drone in the inertial frame is denoted as $\xi = [x, y, z]^T$, and its translation velocity is represented by $v = [v_x, v_y, v_z]^T$. The angular velocity in the body frame is given by $\Omega = [p, q, r]^T$. The relationship between the Euler rates $\dot{\mu}$ and the angular body rates Ω is expressed by the transformation matrix $W(\mu)$, where $\dot{\mu} = W(\mu)^{-1}\Omega$ such that

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & S_\phi T_\theta & C_\phi T_\theta \\ 0 & C_\phi & -S_\phi \\ 0 & S_\phi/C_\theta & C_\phi/C_\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

where $T_* = \tan(*)$. The states of the FA-Hex are defined as $x = [\xi^T, v^T, \mu^T, \Omega^T]^T$, and the system dynamics are expressed as follows:

$$\dot{\xi} = v \quad (1)$$

$$m\ddot{\xi} = mg + R(\mu) \sum_{i=1}^6 F_{B_i} \quad (2)$$

$$I_B \dot{\Omega} = \sum_{i=1}^6 \tau_{B_i} - \Omega \times I_B \Omega \quad (3)$$

Here, $\times$ is the cross product operator, whereas $F_{B_i} = [F_{x_i} \ F_{y_i} \ F_{z_i}]^T$ and $\tau_{B_i} = [\tau_{x_i} \ \tau_{y_i} \ \tau_{z_i}]^T$ represent the forces and torques exerted by the i -th motor expressed in the body frame, respectively.

F_{B_i} and τ_{B_i} depend on the propeller's angular speed ω_i , as well as the motor's orientation α_i within the drone's geometric frame, and the tangential orientation β defined as the angle between the motor's axis z_{m_i} and the drone's z_B axis. This is represented in Figure 2, where $F_i = \sqrt{F_{x_i}^2 + F_{y_i}^2}$ is the projection of F_{B_i} in the (x_B, y_B) plane. The expressions of F_{B_i} and τ_{B_i} are given in Table 1. It can be checked that $F_{B_i} = F_{thrust_i}$, where $\|F_{thrust_i}\| = k_t \omega_i^2$, while $\tau_{B_i} = \tau_{thrust_i} + \tau_{drag_i}$, where $\|\tau_{thrust_i}\| = k_t l \omega_i^2$ and $\|\tau_{drag_i}\| = k_q \omega_i^2$. k_t , k_q and l stand for the thrust coefficient, the drag coefficient, and the distance between the i -th motor and the drone's center of mass respectively.

2.2 FA-Hex Modeling with Cable Suspended Pendulum

The FA-Hex system with the cable-suspended payload operates in a hybrid mode, which relies on the tension force in the cable, as discussed in [14]. The payload's behavior is modeled under two main scenarios: taut and slack cables. The transition between these modes adds intricacy to the modeling approach, thus augmenting the complexity of the optimization function. Nevertheless, by introducing a linear complementary constraint into the optimization problem, it is feasible to eliminate the necessity for the hybrid model [15].

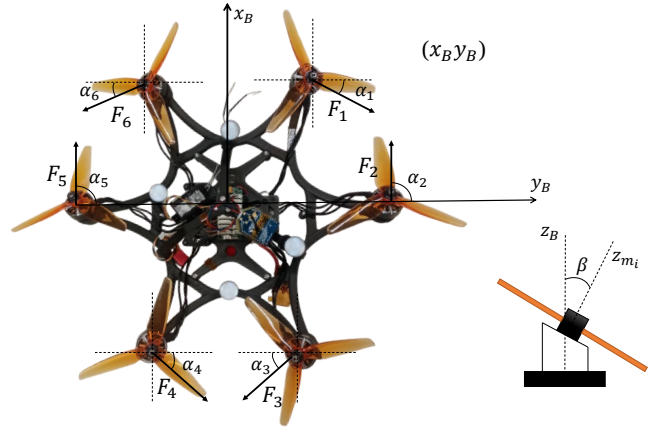


Figure 2: Forces projection in the lateral plane with the motor tilting angles β between z_B and z_{m_i} the axis of the i -th motor.

$F_{x_1} = -k_t S_\beta S_{\alpha_1} \omega_1^2$	$\tau_{x_1} = (-k_t C_\beta S_{\alpha_1} l + k_q S_\beta S_{\alpha_1}) \omega_1^2$
$F_{y_1} = k_t S_\beta C_{\alpha_1} \omega_1^2$	$\tau_{y_1} = (k_t C_\beta C_{\alpha_1} l - k_q S_\beta C_{\alpha_1}) \omega_1^2$
$F_{z_1} = -k_t C_\beta \omega_1^2$	$\tau_{z_1} = (k_t S_\beta l - k_q C_\beta) \omega_1^2$
$F_{x_2} = k_t S_\beta S_{\alpha_2} \omega_2^2$	$\tau_{x_2} = (-k_t C_\beta S_{\alpha_2} l + k_q S_\beta S_{\alpha_2}) \omega_2^2$
$F_{y_2} = k_t S_\beta C_{\alpha_2} \omega_2^2$	$\tau_{y_2} = (k_t C_\beta C_{\alpha_2} l + k_q S_\beta C_{\alpha_2}) \omega_2^2$
$F_{z_2} = -k_t C_\beta \omega_2^2$	$\tau_{z_2} = (-k_t S_\beta l + k_q C_\beta) \omega_2^2$
$F_{x_3} = -k_t S_\beta S_{\alpha_3} \omega_3^2$	$\tau_{x_3} = (-k_t C_\beta S_{\alpha_3} l + k_q S_\beta S_{\alpha_3}) \omega_3^2$
$F_{y_3} = -k_t S_\beta C_{\alpha_3} \omega_3^2$	$\tau_{y_3} = (-k_t C_\beta C_{\alpha_3} l + k_q S_\beta C_{\alpha_3}) \omega_3^2$
$F_{z_3} = -k_t C_\beta \omega_3^2$	$\tau_{z_3} = (k_t S_\beta l - k_q C_\beta) \omega_3^2$
$F_{x_4} = -k_t S_\beta S_{\alpha_4} \omega_4^2$	$\tau_{x_4} = (k_t C_\beta S_{\alpha_4} l - k_q S_\beta S_{\alpha_4}) \omega_4^2$
$F_{y_4} = k_t S_\beta C_{\alpha_4} \omega_4^2$	$\tau_{y_4} = (-k_t C_\beta C_{\alpha_4} l + k_q S_\beta C_{\alpha_4}) \omega_4^2$
$F_{z_4} = -k_t C_\beta \omega_4^2$	$\tau_{z_4} = (-k_t S_\beta l + k_q C_\beta) \omega_4^2$
$F_{x_5} = k_t S_\beta S_{\alpha_5} \omega_5^2$	$\tau_{x_5} = (k_t C_\beta S_{\alpha_5} l - k_q S_\beta S_{\alpha_5}) \omega_5^2$
$F_{y_5} = k_t S_\beta C_{\alpha_5} \omega_5^2$	$\tau_{y_5} = (k_t C_\beta C_{\alpha_5} l - k_q S_\beta C_{\alpha_5}) \omega_5^2$
$F_{z_5} = -k_t C_\beta \omega_5^2$	$\tau_{z_5} = (k_t S_\beta l - k_q C_\beta) \omega_5^2$
$F_{x_6} = -k_t S_\beta S_{\alpha_6} \omega_6^2$	$\tau_{x_6} = (k_t C_\beta S_{\alpha_6} l - k_q S_\beta S_{\alpha_6}) \omega_6^2$
$F_{y_6} = -k_t S_\beta C_{\alpha_6} \omega_6^2$	$\tau_{y_6} = (k_t C_\beta C_{\alpha_6} l - k_q S_\beta C_{\alpha_6}) \omega_6^2$
$F_{z_6} = -k_t C_\beta \omega_6^2$	$\tau_{z_6} = (-k_t S_\beta l + k_q C_\beta) \omega_6^2$

Table 1: Forces and Torques Projections

The state vector of the model is expanded and modified to incorporate both the position and velocity of the load. This state is denoted as $x = [\xi^T, v^T, \mu^T, \Omega^T, \xi_L^T, v_L^T]^T$, where $\xi_L = [x_L, y_L, z_L]^T$ represents the load's position in the inertial frame, and $v_L = [v_{L_x}, v_{L_y}, v_{L_z}]^T$ represents the load's velocity. With the presence of the load, a new tension force T comes into play, affecting the drone's dynamics. Furthermore, due to the cable's rotation around the inertial axis, a unit vector describing a rotation $R_L(\eta)$ is introduced to take account of the drone's orientation in relation to the load. Here, $\eta = [\alpha_L, \beta_L]^T$, as depicted in Figure 1.

$$R_L(\eta) = \begin{bmatrix} \sin(\alpha_L) \cos(\beta_L) \\ \sin(\alpha_L) \sin(\beta_L) \\ \cos(\alpha_L) \end{bmatrix}$$

The extended dynamic model is then represented as follows:

$$\dot{\xi} = v \quad (4)$$

$$(m + m_L)\ddot{\xi} = (m + m_L)g + R(\mu) \sum_{i=1}^6 F_{B_i} + TR_L(\eta) \quad (5)$$

$$I_B \dot{\Omega} = \sum_{i=1}^6 \tau_{B_i} - \Omega \times I_B \Omega \quad (6)$$

$$\xi_L = \xi + R_L(\eta)L \quad (7)$$

$$\dot{\xi}_L = v_L \quad (8)$$

$$\ddot{\xi}_L = m_L g - TR_L(\eta) \quad (9)$$

where T is the tension in the cable, L is the length of the cable, and m_L is the mass of the load respectively.

3 TRAJECTORY OPTIMIZATION PROBLEM

Optimization methods are employed to address trajectory optimization problems by minimizing a cost function, while taking into account different constraints. To achieve a viable and physically relevant solution, the optimization problem needs to encompass the system dynamics along with the practical restrictions on the states and inputs. Broadly, optimization problems fall into two primary categories: direct methods [18] and indirect methods [16].

The indirect method involves formulating the optimization problem as a boundary value problem (BVP) and solving it by defining the co-state vector. It is worth mentioning that the model of the system is incorporated into the Hamiltonian-Jacobian-Bellman equation, which is solved using the BVP approach. On the other hand, the direct optimization problem takes into account a discrete dynamical model along with its constraints. The main distinction between the two methods lies in the integration aspect. Direct methods are known for incorporating integrated solutions for the dynamical model within their approach. It is worth noting that indirect methods generally tend to be more accurate than direct methods, provided that a good initial guess can be obtained, which can be challenging in practice. However, in the realm of robotics applications, direct methods are widely utilized for designing optimal controllers. This preference is largely attributed to their expansive region of convergence [19].

In this work, our focus is on utilizing a direct method to obtain the optimal trajectory. Direct methods can be formulated using shooting and collocation methods, with the choice depending on the specific dynamical equation. When an explicit model is available, the shooting method is often employed. This method involves decomposing the trajectory into sub-optimal intervals and calculating the spline between each interval. The formulation can be done through single shooting, where only the state variables are optimized, or multiple shooting, where both control inputs and states are optimized at each control interval. The resulting discrete problem is then addressed using non-linear programming methods [18].

3.1 General Problem Formulation

We pose our trajectory optimization problem as follows :

$$\begin{aligned} \min_{x,u} \quad & J(x,u) \\ \text{s.t.} \quad & g(x,u) = 0 \quad f(x,u) \geq 0 \end{aligned} \quad (10)$$

In this formulation, x represents the state vector of the system, and u denotes the angular speed of each motor. The equality constraints $g(x,u)$ encompass various constraints, including the dynamical constraint of the system, while the inequality constraints $f(x,u)$ define additional inequality restrictions. To solve the optimization problem, we divide the trajectory into N control sub-intervals. For this purpose, we employ the direct multiple shooting optimization formulation, which offers several advantages over other techniques [18].

3.2 FA-Hex

First, we will define the optimization problem for the FA-Hex targeting photography application.

- a) *Cost Function*: The cost function to be minimized is defined as follows:

$$\min_x J = S_f t_f + \int_0^{t_f} x^T Q x dt \quad (11)$$

Here, $x = [\xi^T, v^T, \mu^T, \Omega^T]^T$ and $Q \in \mathcal{R}^{12 \times 12}$ represents the weight cost assigned to the states of the system, t_f is the time allotted to the whole scenario, and S_f denotes the slack weight used to handle the priority of the specified time.

- b) *Dynamical Model Constraint*: The dynamical model, defined in Section 2.1, is discretized using fourth-order Runge-Kutta. This model is employed to ensure the attainment of dynamically feasible states by solving $x_{k+1} = h(x_k, u_k)$, where h is a vector field representing the discretized dynamical model of the system.
- c) *Multiple-Shooting Constraint*: The method involves computing the next state using the discretized model. The computed next state is constrained to align with the optimized parameter, which can be achieved by ensuring that $x_{\text{next}} - x_k = 0$. Here, x_k represents the system state at the k -th interval, and x_{next} corresponds to the state calculated using the discretized dynamical model.
- d) *Boundary Conditions*: To ensure that the optimization problem begins and ends within feasible constraints, additional boundary constraints are included. The variables x_{initial} and x_{final} are defined and incorporated into the equality constraint $g(x,u)$ as $x_0 - x_{\text{initial}} = 0$ and $x_N - x_{\text{final}} = 0$.

- e) *State and Input Constraint*: The states and inputs of the system should not exceed their respective boundaries. To enforce this requirement, two additional equations (12, 13) are introduced to the function $f(x, u)$.

$$x_{min} \leq x_k \leq x_{max} \quad (12)$$

$$u_{min} \leq u_k \leq u_{max} \quad (13)$$

- f) *Waypoint Navigation Constraint*: The primary objective of this work is to navigate and fly over specific points. To achieve this goal, an additional constraint is introduced: $w_p - \delta < x_k < w_p + \delta$, where w_p represents the specified waypoint. The inclusion of δ in this equation ensures the robustness of the algorithm and helps avoid excessive equality constraints. It is important to note that this problem does not aim to optimize time since such optimization conflicts with the defined constraint. For more detailed information on time-optimal trajectory optimization the reader can refer to [20].

3.3 FA-Hex With Cable Suspended Pendulum

In this section, we will introduce the extended constraints and objective function that are used to optimize the trajectory for FA-Hex with a cable-suspended pendulum.

- a) *Cost Function*: The extended cost function to be minimized on the variables is defined as follows:

$$\min_{x, T, L} J = S_f t_f + \int_0^{t_f} x_{ext}^T Q_L x_{ext} dt + \int_0^{t_f} (\bar{R}_0 T + \bar{R}_1 (L_0 - L)) dt \quad (14)$$

Here, $x_{ext} = [\xi^T, v^T, \mu^T, \Omega^T, \xi_L^T, v_L^T]^T$, $Q_L \in \mathcal{R}^{18 \times 18}$ and the added terms $\bar{R}_0 T$ and $\bar{R}_1 (L_0 - L)$ represent the cost associated with the taut and slack modes of the cable, where $\bar{R}_0$ and $\bar{R}_1$ are scalar weights, and L_0 is the initial length of the load in taut mode. These terms are included in the objective function to address the hybrid modeling problem of the pendulum. The trajectory is generated with a corresponding slack and taut mode, based on the weighting values.

- b) *Dynamical Model Constraint*: The extended dynamical model introduced in Section 2.2 is also discretized using fourth-order Runge-Kutta and incorporated into the equality constraints $g(x, u)$.
- c) *Linear Complementary Constraint*: This constraint is added to eliminate the hybrid dynamical model, assuming that either the slack mode or the taut mode is present. It is represented by $T(L_0 - L)$, where L_0 is the cable length when taut, and T represents the tension in the cable.

- d) *Tension Boundaries Constraint*: To ensure a realistic and feasible solution, it is important to impose bounds on the tension. This can be achieved by incorporating the constraint $0 \leq T \leq T_{max}$, where T_{max} represents the maximum allowable tension. The value of T_{max} can be determined based on prior information or known limitations of the system.

- e) *Cable Length Constraint*: In addition to the complementary constraint, the length of the cable should be positive, $L \geq 0$.

- f) *Payload Swung Angle Constraint*: To prevent swinging and potential collisions with the drone, a constraint is introduced as $\alpha_L \geq \alpha_{Lmin}$ and $\alpha_L \leq \alpha_{Lmax}$.

The other constraints defined in the FA-Hex drone are primarily utilized as constraints for this particular application.

4 SIMULATIONS

The defined optimization problem is formulated as a non-linear optimization problem using the CasADi Toolbox [21]. As mentioned earlier, this study considers two primary scenarios: simple FA-Hex and FA-Hex with a cable-suspended pendulum. A comparison between FA-Hex and UA-Hex is accomplished showing the enhancing of maneuverability of the FA-Hex. The drone parameters used to simulate the drone model are presented in Table 2.

$m = 0.6656 \text{ kg}$	$l = 0.15 \text{ m}$	$g = 9.81 \text{ m/sec}^2$
$k_t = 3.4 \times 10^{-5} \text{ N}/\omega^2$	$k_q = 3.4 \times 10^{-6} \text{ Nm}/\omega^2$	$I_{xx} = 0.0411 \text{ kgm}^2$
$I_{yy} = 0.0478 \text{ kgm}^2$	$I_{zz} = 0.0599 \text{ kgm}^2$	$\alpha_1 = \frac{\pi}{6} \text{ rad}$
$\alpha_2 = \frac{\pi}{2} \text{ rad}$	$\alpha_3 = \frac{\pi}{6} \text{ rad}$	$\alpha_4 = \frac{\pi}{6} \text{ rad}$
$\alpha_5 = \frac{\pi}{2} \text{ rad}$	$\alpha_6 = \frac{\pi}{6} \text{ rad}$	$\beta = \frac{\pi}{8} \text{ rad}$
$m_L = 0.05 \text{ kg}$		

Table 2: Drone Simulation Parameters

4.1 FA-Hex and UA-Hex Trajectory Generation

The scenario involves the drone passing through multiple waypoints. To assess the algorithm's ability to achieve minimal banking angles, several test cases were conducted. A trajectory comprising four points is shown in Figure 4. The time allotted for this scenario is $t_f = 8 \text{ s}$, with $N = 100$ control intervals. It's worth noting that, as classified by [5], FA vehicles belong to the category of Lateral Bounded Force vehicles (LBF). In these vehicles, lateral acceleration is constrained to be smaller than the vertical acceleration of the drone. The UA-Hex shares the same parameters as the FA-Hex, except for having a tilting rotor angle $\beta = 0$. The achieved velocity of the FA-Hex is approximately 1.5 m/s when flying without banking, while the velocity of the UA-Hex can reach 2.5 m/s . Figure 3 displays the optimized Euler angles for both the UA-Hex and FA-Hex in the considered scenario. Clearly, based on the optimized trajectory, the FA-Hex maintains banking angles close to zero degrees while striving to achieve the same trajectory.

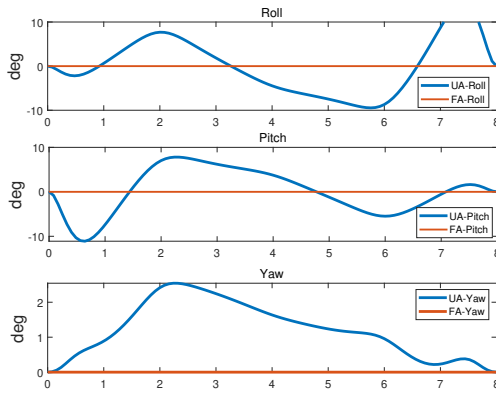


Figure 3: Euler angles comparison between FA-Hex and UA-Hex.

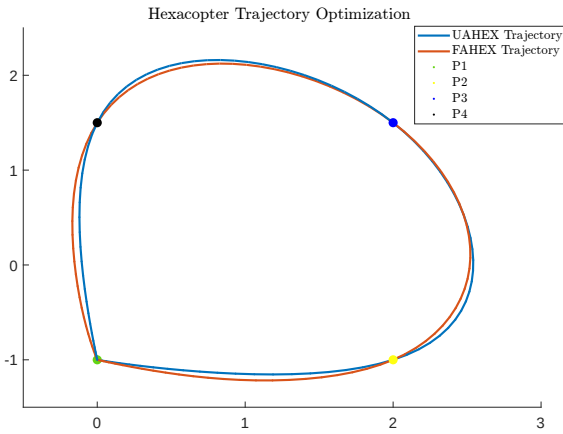


Figure 4: Projection in the horizontal plane of the optimized trajectory for FA-Hex and UA-Hex drones as they fly between specified points and then return to their initial positions. The points are defined by their coordinates: $P1 = [0, -1, 1.3]$, $P2 = [2, -1, 1.3]$, $P3 = [2, 1.5, 1.8]$ and $P4 = [0, 1.5, 1.8]$.

4.2 FA-Hex With Cable Suspended Pendulum

In another application, the optimization problem is expanded, as explained in Section 3.3, to be solved across the optimization variables x, u, T, L , and q . As in Section 4.1, the scenario involves flying between predetermined points. Utilizing linear complementary constraints, it becomes feasible to generate a trajectory with minimal slack, indicating that the cable length remains close to L_0 for the majority of the trajectory. The simulation results are depicted in Figure 5, illustrating both the payload and drone trajectories.

4.3 Initial Guess

The initial guess plays a pivotal role in ensuring the optimization problem converges to its optimal solution. Providing a feasible initial guess that closely approximates the expected solution is essential. This step is critical for achiev-

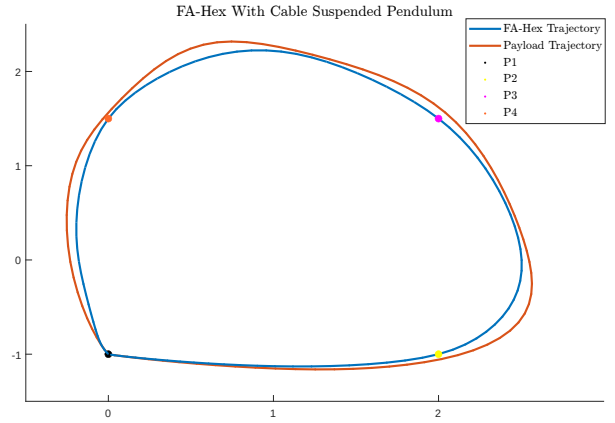


Figure 5: Projection in the horizontal plane of the optimized trajectory for FA-Hex with cable suspended pendulum.

ing convergence and reducing computational time needed to solve the problem. In this context, it is important to note that the drone is initially treated as a UA system. The differential flatness approach is employed to generate an initial guess for the problem. Furthermore, motion planning algorithms grounded in graph theory, such as the A^* and RTT algorithms, can be employed. These algorithms facilitate the creation of a global path, effectively aiding in solving the trajectory planning problem.

5 CONCLUSION AND EXTENSIONS

In this work, we have proposed a trajectory optimization formulation for FA-Hex systems. The simulations demonstrate the feasibility of this approach by generating trajectories with minimal banking angles. However, the algorithms used in this study have certain constraints when it comes to defining the nodes and time. The algorithm shows fast performance when the trajectory is divided into $N = 100$ control intervals, taking only a few seconds to converge to the optimal solution. Future work could involve extending the problem to incorporate avoidance of static defined obstacles by implementing specific maneuvers.

As an extension of this paper, we are currently working on the identification and control of the real FA-Hex system. Our objective is to conduct real experimental flights to test the optimized trajectory in the mentioned scenarios. The planned tests will utilize the Paparazzi autopilot [22] with a modified INDI controller or feedback linearization.

REFERENCES

- [1] Pedro J. Sanchez-Cuevas, Antonio Gonzalez-Morgado, Nicolas Cortes, Diego B. Gayango, Antonio E. Jimenez-Cano, Anibal Ollero, and Guillermo Heredia. Fully-actuated aerial manipulator for infrastructure contact inspection: Design, modeling, localization, and control. *Sensors*, 20(17), 2020.

- [2] Blake Chamberlain and Waseem Sheikh. Design and implementation of a quadcopter drone control system for photography applications. In *Intermountain Engineering, Technology and Computing (IETC)*, pages 1–7, 2022.
- [3] Mahmoud Hamandi, Federico Usai, Quentin Sablé, Nicolas Staub, Marco Tognon, and Antonio Franchi. Design of multirotor aerial vehicles: A taxonomy based on input allocation. *The International Journal of Robotics Research*, 40:027836492110259, 2021.
- [4] Ramy Rashad, Jelmer Goerres, Ronald Aarts, Johan B. C. Engelen, and Stefano Stramigioli. Fully actuated multirotor UAVs: A literature review. *IEEE Robotics Automation Magazine*, 27(3):97–107, 2020.
- [5] Antonio Franchi, Ruggero Carli, Davide Bicego, and Markus Ryll. Full-pose tracking control for aerial robotic systems with laterally bounded input force. *IEEE Transactions on Robotics*, 34(2):534–541, 2018.
- [6] Markus Ryll, Giuseppe Muscio, Francesco Pierri, Elisabetta Cataldi, Gianluca Antonelli, Fabrizio Caccavale, and Antonio Franchi. 6D physical interaction with a fully actuated aerial robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5190–5195, 2017.
- [7] Sujit Rajappa, Markus Ryll, Heinrich H. Bühlhoff, and Antonio Franchi. Modeling, control and design optimization for a fully-actuated hexarotor aerial vehicle with tilted propellers. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4006–4013, 2015.
- [8] Marcin Odelga, Paolo Stegagno, and Heinrich H. Bühlhoff. A fully actuated quadrotor UAV with a propeller tilting mechanism: Modeling and control. In *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 306–311, 2016.
- [9] Jorge M. Arizaga, Herman Castañeda, and Pedro Castillo. Adaptive control for a tilted-motors hexacopter UAS flying on a perturbed environment. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 171–177, 2019.
- [10] Jorge M. Arizaga, Herman Castañeda, and Pedro Castillo. Payload swing attenuation of a fully-actuated hexacopter via extended high gain observer based adaptive sliding control. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 901–908, 2021.
- [11] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *IEEE International Conference on Robotics and Automation*, pages 2520–2525, 2011.
- [12] Markus Hehn and Raffaello D’Andrea. Quadcopter trajectory generation and control. *IFAC Proceedings Volumes*, 44(1):1485–1491, 2011. 18th IFAC World Congress.
- [13] Jun Zeng, Prasanth Kotaru, Mark W. Mueller, and Koushil Sreenath. Differential flatness based path planning with direct collocation on hybrid modes for a quadrotor with a cable-suspended payload. *IEEE Robotics and Automation Letters*, 5(2):3074–3081, 2020.
- [14] Koushil Sreenath and Taeyoung Lee. Geometric control and differential flatness of a quadrotor UAV with a cable-suspended load. pages 2269–2274, 2013.
- [15] Davide Falanga, Philipp Foehn, Davide Scaramuzza, Naveen Kuppaswamy, and Russ Tedrake. Fast trajectory optimization for agile quadrotor maneuvers with a cable-suspended payload. 2017.
- [16] Danial Hashemi and Hamidreza Heidari. Trajectory planning of quadrotor UAV with maximum payload and minimum oscillation of suspended load using optimal control. *Journal of Intelligent & Robotic Systems*, 100(3-4):1369–1381, 2020.
- [17] Nikhil Potdar, Guido Croon, and Javier Alonso-Mora. Online trajectory planning and control of a MAV payload system in dynamic environments. *Autonomous Robots*, 44:1065–1089, 2020.
- [18] Michael Posa and Russ Tedrake. *Direct Trajectory Optimization of Rigid Body Dynamical Systems through Contact*, pages 527–542. 2013.
- [19] Ricard Bordalba, Tobias Schoels, Lluís Ros, Josep Porta, and Moritz Diehl. Direct collocation methods for trajectory optimization in constrained robotic systems. *IEEE Transactions on Robotics*, 32(1):183–202, 2022.
- [20] Philipp Foehn, Angel Romero, and Davide Scaramuzza. Time-optimal planning for quadrotor waypoint flight. *Science robotics*, 6(56), 2021.
- [21] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.
- [22] Michel Gorraz Gautier Hattenberger, Murat Bronz. Using the paparazzi UAV system for scientific research. *International Micro Air Vehicle Conference and Competition (IMAV)*, pages 247–252, 2014.

Rigid Airborne docking between a fixed-wing UAV and an over-actuated multicopter

J.J.E. Laffita van den Hove*, E.J.J. Smeur, and B.D.W. Remes

Delft University of Technology, Faculty of Aerospace Engineering, Kluyverweg 1, Delft, Zuid-Holland, The Netherlands

ABSTRACT

Fixed-wing aircraft fly longer, faster, and further than rotorcraft, but cannot take off or land vertically. Hybrid drones combine VTOL with a wing for forward flight, but the hovering system generally makes them less efficient than a pure fixed-wing. We propose an alternative, in which a rotorcraft is used to assist the fixed-wing UAV with the VTOL portions of the flight. This paper takes the first steps towards this alternative by developing and testing an overactuated rotorcraft that can autonomously dock onto a target at fixed-wing velocities. The control system uses Incremental Non-Linear Dynamic Inversion Control (INDI) to achieve linear accelerations with lateral and longitudinal motors, enabling robust horizontal control independent of attitude. A relative guidance algorithm for the docking approach path is presented, along with a vision sensing approach using ArUco markers and IR LEDs. Successful docking and separation were achieved in the wind tunnel at speeds of up to 15m/s.

1 INTRODUCTION

Unmanned aerial vehicles have seen a rapid increase in use and development over the last decade[1]. For most uses, it is more common to use rotorcraft as opposed to fixed-wing aircraft, due to their capability of vertical takeoff and landing (VTOL) as well as hovering. While the need to hover may be a pre-requisite for some missions, it is not a requirement for all of them. Of course being able to takeoff and land vertically is beneficial, but it comes at the cost of flight endurance and range[2].

Hybrid UAVs combine VTOL capability with a wing for efficient forward flight, but they tend to be harder to control during hovering, due to the wing surface that can catch gusts, and they are less efficient during cruise flight due to the extra systems needed to facilitate hovering. That is where the need for airborne docking comes in. A rotorcraft could be used to assist in the takeoff and landing of the fixed-wing aircraft, such that the fixed-wing can be optimized for efficient flight while retaining the VTOL capability. The authors propose the

use of an overactuated rotorcraft with electromagnets as the docking solution.

1.1 Relevant work

Airborne docking systems can be categorized as rigid or non-rigid, the former being systems for which the docking apparatus is comprised of rigid members while the latter utilizes flexible materials that can easily deform such as ropes or nets.

Various non-rigid docking solutions have been developed to recover fixed-wing UAVs without a landing strip. One approach involves arresting systems using horizontal or vertical ropes in combination with a hooking mechanism at the tail or wingtips [3, 4, 5]. This can be achieved using a multicopter, a crane, or a kite dragged by a moving vehicle. Although these approaches require infrastructure and manual recovery, the control of the fixed-wing does not need to be very precise in order to attach to the rope.

A completely airborne variation of the aforementioned solution is presented in [6], where two multicopters are used to carry the arresting rope. The rotorcrafts used RTK GPS to position themselves in the right position for successful capture. The fixed-wing has a line with a hook attached to it, which it uses to hook onto the horizontal catch line held by the rotorcrafts. The recovery margins are dictated by the length of the hook line and horizontal catch line, which can potentially be increased. A similar method was used with a net in [7], which was experimentally verified [8]. However, the net method had drawbacks including increased weight and wind influence, as well as a higher risk of equipment damage compared to single hook alternatives.

Current systems that would allow for docking and separation mid-flight have been in the form of a probe and drogue approach. Extensive research has gone into Air-to-Air refueling for large aircraft, which is summarized in [9]. In the world of smaller UAVs, Wilson successfully had a fixed-wing UAV to autonomously dock with a drogue towed by another UAV[10]. To improve the estimation of the relative distance between the UAVs, his system fuses direct vehicle-to-vehicle observations in the state estimates. Dynetics successfully recovered an X-61A aircraft using a drogue towed from a C130 Aircraft[11].

With regards to rigid solutions, Steven Lukow aimed to design and fabricate a prototype of an autonomous multirotor drone that can dock onto a larger fixed-wing drone for Mars exploration [12]. Electromagnetic coils mounted on the

*Email address(es): jonathaslvdh@gmail.com

fixed-wing drone were used to form a connection with two neodymium magnets mounted on the top of the multirotor. A demonstration was not shown. In [13] the author uses a marker on an unmanned ground vehicle (UGV), such that a fixed-wing UAV could recognize the target and relative pose, and finally land on it. Tests were done at very slow speeds, and the system became unstable at higher speeds. In [14] the same setup is used, but this time with a gimbaled camera that points towards the received location of the ground vehicle. Successful landings were achieved on a high-speed ground vehicle traveling at 50km/h.

Clearly, there have been many advancements toward airborne docking. However, there is no instance of it being done between rotorcraft and a fixed-wing UAV at forward flight velocities.

1.2 Contributions

The airborne docking concept proposed involves an over-actuated multicopter that approaches a fixed-wing aircraft from below to avoid disturbing the airflow over the aircraft's aerodynamic surfaces and dock onto it rigidly using electromagnets. To achieve the precise relative control desired for the docking maneuver, three additional motors are added to the multicopter: two laterally and one longitudinally. This also allows to decouple position from attitude.

This paper focuses on the rotorcraft, and the contributions are as follows. A relative control and guidance strategy for the overactuated platform is put forward, and validated by performing wind tunnel experiments at fixed-wing velocities with a static target. The use of electromagnets as a docking system is also put to the test. A relative sensing method that uses IR LEDs in combination with retroreflective ArUco markers are used. An illustration of the system can be seen in Figure 1.

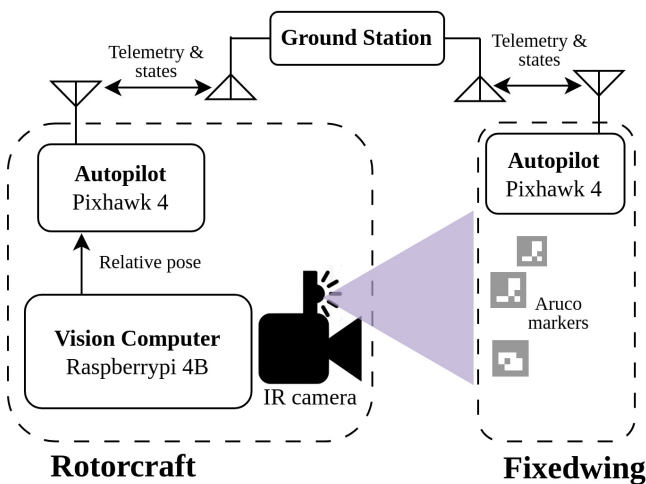


Figure 1: System Architecture.

2 CONTROL

2.1 INDI for linear accelerations

Traditional rotorcraft need to adjust the attitude in order to control their position. This is identified as a potential problem in the docking process. To be able to control the position of the rotorcraft independently of attitude, one back propeller and two side propellers are used, resulting in an over-actuated platform, as can be seen in Figure 2. To simplify the control problem, the four propellers facing downward are used for attitude control and the generation of vertical lift, while the 3 side propellers are used for horizontal position control. The attitude of the drone is controlled with INDI [15]. It is assumed that the moments resulting from the lateral-acting motors are negligible, and any effect they do have will be treated as a disturbance. For position control, a variation of [16] is implemented, where the side actuation is used instead of the attitude angle to achieve horizontal control. The two reference frames that are used in this section are the body frame and the North, East Down (NED) frame, which is assumed to be an inertial frame. The body frame is depicted in Figure 2 and is denoted by subscript B . The NED frame, indicated with subscript N , has its axes pointing North, East, and Down, and its origin is fixed to a point on earth of choice.

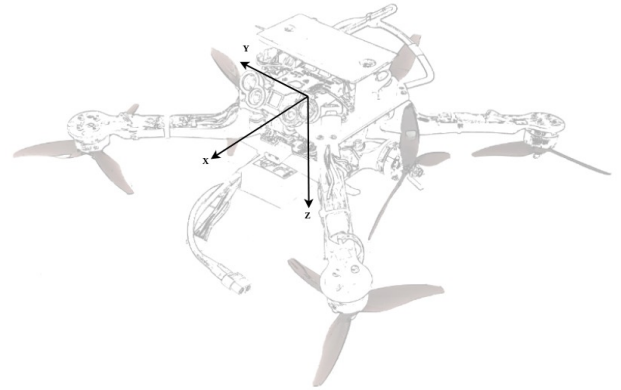


Figure 2: Body reference Frame.

The position dynamics of the aircraft in the NED frame is given by Newton's second law of motion:

$$\ddot{\xi} = g + \frac{1}{m}F(\dot{\xi}, w) + \frac{1}{m}T_N(\eta, T_B), \quad (1)$$

where ξ is the position of the vehicle, g is the gravity vector, F is the sum of aerodynamic forces acting on the airframe, and T is the thrust vector, which contrary to other conventional multicopters does not only act through the z -axis of the vehicle.

The aerodynamic force is a function of aircraft velocity $\dot{\xi}$ and wind vector w , and the thrust vector is dependent on the attitude of the drone $\eta = [\phi \ \theta \ \psi]^T$, expressed in Euler angles with the traditional ZYX rotation order, as well as the

total thrust T_B in the body frame. Due to the side motors, the vehicle can produce thrust along all axes, as opposed to only along the Z axis. T_N is given by:

$$T_N(\eta, T_B) = M_{NB}(\eta)T_B = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi s\phi & s\psi s\theta + c\phi s\theta c\phi \\ s\psi c\theta & c\psi s\theta s\phi & s\psi s\theta c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \cdot \begin{bmatrix} T_{BX} \\ T_{BY} \\ T_{BZ} \end{bmatrix}, \quad (2)$$

where the sine and cosine are abbreviated by the letters s and c respectively. Now a first order Taylor expansion can be done on Equation 1 to obtain Equation 3:

$$\begin{aligned} \ddot{\xi} = & g + \frac{1}{m}F(\xi_0, w_0) + \frac{1}{m}T_N(\eta_0, T_{B0}) \\ & + \frac{\sigma}{\sigma\xi} \frac{1}{m}F(\xi, w_0)|_{\xi=\xi_0}(\xi - \xi_0) \\ & + \frac{\sigma}{\sigma w} \frac{1}{m}F(\xi_0, w)|_{w=w_0}(w - w_0) \\ & + \frac{\sigma}{\sigma\phi} \frac{1}{m}T_N(\phi, \theta_0, \psi_0, T_{BX0}, T_{BY0}, T_{BZ0})|_{\phi=\phi_0}(\phi - \phi_0) \\ & + \frac{\sigma}{\sigma\theta} \frac{1}{m}T_N(\phi_0, \theta, \psi_0, T_{BX0}, T_{BY0}, T_{BZ0})|_{\theta=\theta_0}(\theta - \theta_0) \\ & + \frac{\sigma}{\sigma\psi} \frac{1}{m}T_N(\phi_0, \theta_0, \psi, T_{BX0}, T_{BY0}, T_{BZ0})|_{\psi=\psi_0}(\psi - \psi_0) \\ & + \frac{\sigma}{\sigma T_{BX}} \frac{1}{m}T_N(\phi_0, \theta_0, \psi_0, T_{BX}, T_{BY0}, T_{BZ0})|_{T_{BX}=T_{BX0}}(T_{BX} - T_{BX0}) \\ & + \frac{\sigma}{\sigma T_{BY}} \frac{1}{m}T_N(\phi_0, \theta_0, \psi_0, T_{BX0}, T_{BY}, T_{BZ0})|_{T_{BY}=T_{BY0}}(T_{BY} - T_{BY0}) \\ & + \frac{\sigma}{\sigma T_{BZ}} \frac{1}{m}T_N(\phi_0, \theta_0, \psi_0, T_{BX0}, T_{BY0}, T_{BZ})|_{T_{BZ}=T_{BZ0}}(T_{BZ} - T_{BZ0}) \end{aligned} \quad (3)$$

The first terms $g + \frac{1}{m}F(\xi_0, w_0) + \frac{1}{m}T_N(\eta_0, T_{B0})$ can be simplified to the acceleration at the previous time step. This can be obtained by using the IMU acceleration readings and rotating them to the NED frame. The gravitational acceleration must then be added to that. There is no good estimate for the partial derivatives of $F(\xi, w)$ as developing an aerodynamic model would be very time-consuming and the changes in the wind are hard to obtain. These two terms will therefore be taken as 0. This does not mean these forces are neglected, as they are included in the term ξ_0 . The partial derivatives of ϕ , θ , and ψ will also be neglected, as for the time being, there will not be any aggressive attitude commands in the airborne docking maneuver. Implementation of these terms is considered future work.

Equation 3 can now be rewritten into:

$$\ddot{\xi} = \ddot{\xi}_0 + \frac{1}{m}G(\eta_0)(u - u_0), \quad (4)$$

where $u = [T_{BX} \quad T_{BY} \quad T_{BZ}]$ and

$$G(\eta_0, T_{B0}) = R_{NB} \cdot \begin{bmatrix} \frac{dT_{BX}}{dT_{BXc}} \\ \frac{dT_{BX}}{dT_{BYc}} \\ \frac{dT_{BY}}{dT_{BYc}} \\ \frac{dT_{BZ}}{dT_{BZc}} \end{bmatrix}, \quad (5)$$

which is referred to as the control effectiveness matrix.

The accelerometer measurements obtained to determine $\ddot{\xi}_0$ are normally very noisy due to vibrations stemming from the rotating propellers. They are therefore filtered using a second order low pass filter. Since a filter introduces a delay, all other terms with subscript 0 must therefore also be filtered with the same filter. Now Equation 4 can be inverted to obtain:

$$u_c = u_f + mG^{-1}(\eta_0)(v_{\ddot{\xi}} - \ddot{\xi}_f). \quad (6)$$

Note that all values that are filtered are denoted with a subscript f . The virtual control $v_{\ddot{\xi}}$ is the desired acceleration of

the vehicle and u_c is the resulting control output. The control block diagram can be seen in Figure 3.

2.1.1 Thrust estimation

The actuator dynamics can be modeled using the 1st order system shown in Equation 7

$$A(z) = \frac{\alpha}{z - (1 - \alpha)} \quad (7)$$

where α was estimated to be 0.069 at 500Hz.

The vertical control increment $\tilde{T}_{BZ}$ is controlled through the four down-facing propellers and is passed to the inner attitude loop. The way this thrust increment is used is further described in [15]. Since the over-actuated multicopter is designed to be flying forward at fixed-wing aircraft speeds, there will always be an aerodynamic force acting in the negative X_B direction. One motor acting forwards is therefore sufficient for longitudinal control. At lower speeds, where aerodynamic drag is not significant for longitudinal control, the two aircraft are already docked, so precise control is not needed. In this case, reversing the direction of rotation of the propeller is sufficient for longitudinal control, albeit using different gains due to the slower dynamics. For lateral control, two motors facing away from each other are used. The PWM command to each lateral motor is:

$$T_{Lc} = T_{min} \leq T_{base} + T_{BYc} \leq T_{max}, \quad (8)$$

$$T_{Rc} = T_{min} \leq T_{base} - T_{BYc} \leq T_{max} \quad (9)$$

Where T_{Lc} and T_{Rc} are the command to the left and right motor respectively, T_{base} is the baseline thrust of both motors and T_{BYc} is the commanded side thrust. The total side thrust T_{BY} then results from subtracting the thrust of the left and right motor.

A quadratic relationship governs the rotational speed of the propellers and thrust. That means that the thrust effectiveness ($\frac{dT}{dT_c}$) of each motor increases linearly with its rotational speed. In the context of the side propellers, increasing T_{BYc} will lead to an increase in the thrust effectiveness of the left motor. However, it will also lead to an equal decrease in the thrust effectiveness of the right motor. The overall effectiveness of the lateral command $\frac{dT_{BY}}{dT_{BYc}}$ will therefore remain constant as long as $T_{BYc} \leq \min(T_{base} - T_{min}, T_{max} - T_{base})$. Beyond that, only the thrust effectiveness of the unsaturated motor will change, so the assumption that the total effectiveness of both motors remains constant does not hold anymore. It will then change linearly for higher lateral commands.

For the longitudinal thrust, the control effectiveness of the pusher motor was determined at an airspeed of 12m/s. Since the effectiveness changes with the propeller RPM, flying at higher or lower speeds tends to give an oscillatory response.

In Figure 4, we can appreciate the vehicle response to lateral acceleration commands at various roll angles. The acceleration is tracked well, except for when saturation occurs due



<http://www.imavs.org/>

<http://www.imavs.org/>



<http://www.imavs.org/>

<http://www.imavs.org/>

<http://www.imavs.org/>

<http://www.imavs.org/>

<http://www.imavs.org/>



<http://www.imavs.org/>

<http://www.imavs.org/>

<http://www.imavs.org/>

<http://www.imavs.org/>

<http://www.imavs.org/>

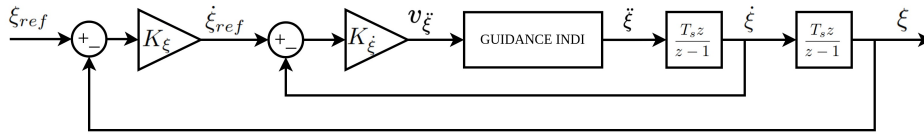


Figure 6: Position control loop

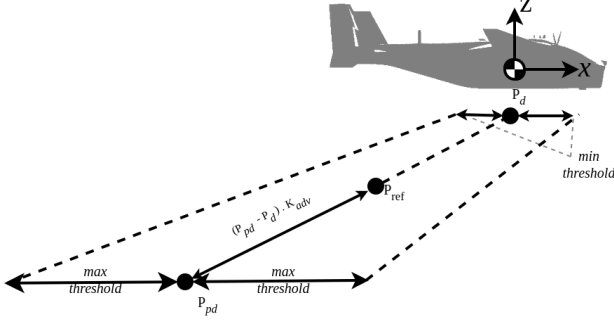


Figure 7: Setpoint generation.

a pitch or roll change of the fixed-wing, but is affected by a change in its heading.

In the docking sequence, the rotorcraft first reaches the *pre-docked* position. Then, the setpoint moves along the approach path towards the *Docked* position. At every iteration of the setpoint generation module (running at 100 Hz), the rotorcraft's horizontal distance to the approach center-line is evaluated, and if it is within the error threshold, then the reference progresses forward. If the aircraft is outside the error threshold, then the reference is moved backward. The amount by which the setpoint progresses along that line is defined by the value k_{adv} .

The error threshold E_{th} is also scaled by k_{adv} , to allow for more error when further away from the fixed-wing. During the experiments, this error threshold decreases linearly from 0.4m at the pre-docked setpoint, to 0.1m at the docked setpoint.

To avoid rapid changes in the reference position due to a momentary change in the heading of the fixed-wing, ψ_{fw} is first filtered with a low pass filter. The commanded heading of the rotorcraft is also set to ψ_{fw} .

2.4 Switching between GPS and vision

To achieve successful docking, the relative position should be accurately estimated. This could be potentially possible with RTK GPS, but in order not to rely on an RTK fix, computer vision is used to augment the GPS. The pre-docked setpoint is, therefore, a point where the two aircraft can be positioned in a safe, yet close enough distance so that the vision system is operative. Once the vision estimates have a good confidence, the relative positioning is done purely on vision. During the approach, vision dropouts can occur. To avoid dropping back to GPS and therefore to the pre-docked setpoint straight away, a buffer is implemented, as can be seen

in algorithm 1 where the *Vision confidence* decreases with every loop iteration during a dropout, eventually reaching 0 and forcing the fallback to GPS. While in the dropout, the relative position estimate remains the same, and therefore, the relative acceleration command also remains the same.

Input: Detected markers, position estimate (vision & GPS)

Output: updated Setpoint

Setpoint $\leftarrow$ predocked Setpoint;

while in flight do

if markers detected > 2 **then**

 | increase vision confidence;

else

 | decrease vision confidence;

end

if vision confidence > threshold **then**

 // Use vision

if horizontal error < threshold **then**

 | move Setpoint forward;

else

 | move Setpoint backwards;

end

else

 // Use GPS

 Setpoint $\leftarrow$ predocked Setpoint;

end

end

Algorithm 1: Setpoint generation at module frequency (100Hz)

3 RELATIVE SENSING

To implement the developed control system, the relative position, relative heading, and relative velocity of the two vehicles must be estimated accurately. For the proposed rigid docking apparatus to work the position accuracy must be at least 5cm. To do so, GPS and a monocular camera will be used in cooperation. The GPS measurements, which can achieve a relative position accuracy lower than 2 meters [17], will be used to reach a formation within the range of the vision system. Once within range, the more accurate vision system estimates the relative pose and its derivative to perform the docking maneuver.

3.1 Vision

To estimate the aircraft's relative pose using a monocular camera, a Perspective-N-Point problem is solved using OpenCV. The process involves detecting known points on the aircraft and finding the transformation matrix which minimizes the re-projection error between the 2D image points and the 3D object points. In [18] the authors used the corners of ArUco markers placed on a drone to estimate relative pose. In [10] Wilson used a known pattern of IR LEDs that could

be used as object points and could be detected at a distance of up to 30m away.

In this proof of concept, we use a combination of both implementations; ArUco markers will be used for their already-implemented detection algorithms, and the markers are made of IR reflective tape (3M High-Reflex-Tape type 7610). The rotorcraft is equipped with 940nm IR LEDs, and a band-pass filter around the same frequency is used on the camera to minimize any unwanted light. This makes the markers clearly visible in a wide range of lighting conditions.

A comparison between a regular black and white marker and the proposed retroreflective markers is shown in Figure 8. The retroreflective marker offers better visibility and distinct edges, enhancing detection. Selection of the reflective tape is critical to prevent overexposure caused by the IR LEDs. The corner points of the ArUco markers are used as the object points. To ensure marker detection at a range, a combination of small and large ArUco markers is placed around the aircraft. This approach increases the scope of the marker detection [18].

The vision computer used is a RaspberryPi 4B that is constantly running three threads. The first thread processes the camera feed and estimates the relative pose (translation and rotation). The second thread uses the same feed to calculate the optical flow. The third thread sends the estimated relative distance in the body frame, the optical flow, and the relative heading to the autopilot through a serial interface. With this setup, the relative pose is estimated at an average of 32Hz at a resolution of 640 x 480 pixels.



Figure 8: Conventional ArUco marker compared to proposed marker when backlit.

3.2 Calculating the relative state

After the visible markers have been detected, the 4 corner points of each of them are known. Given that the marker ID of each of these points are known, and the exact location where these markers are placed on the UAV is also known, we now have a series of (2D) image points with their respective (3D) object points. This data can be directly used in the SolvePnP() function in OpenCV. The estimated translation vector T_{vec} obtained can then be used to find the relative NED position using:

$$\xi_{rel} = -R_{NB}R_{BC}T_{vec}, \quad (11)$$

where R_{NB} is the rotation matrix from the Body frame to the NED frame, and R_{BC} is the rotation matrix from the camera frame to the body frame, which can be refined by comparing the estimated relative position to the true relative position.

$$\dot{\xi}_{rel} = \dot{\xi}_{rc} - \dot{\xi}_{fw}, \quad (12)$$

At any point in time, the rotation matrix from the body frame to the NED frame is calculated using the estimated attitude of the rotorcraft. The relative heading ψ_{rel} can be calculated by applying the Rodrigues rotation formula to the obtained rotation vector. The estimated fixed-wing heading ψ_{fw} then becomes:

$$\psi_{fw} = \psi_{rc} + \psi_{rel}, \quad (13)$$

where ψ_{rc} is the heading of the rotorcraft. This way, the final approach can take place without communication between the vehicles.

The vision distance estimates are compared with measurements of an external infrared tracking system during several docking attempts. It was observed that the vision error decreases as the aircraft approaches the target. After compensating for camera offset angles, which are seen as constant errors that linearly increased with distance, the absolute error of the vision system is shown in Figure 9. As the rotorcraft approaches the target, the error decreases to below 5cm.

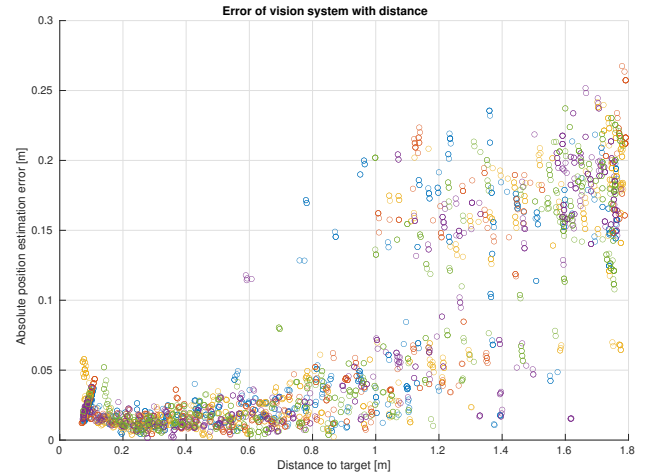


Figure 9: Vision position estimation error with distance.

4 WIND TUNNEL TESTS

As a stepping stone towards outdoor flight, wind tunnel tests were carried out. This allowed us to validate that the over-actuated platform can indeed serve the intended purpose in forward flight, at fixed-wing UAV velocities. The wind tunnel setup allows to keep the follower drone tethered at all times and track its position at mm accuracy, allowing it to be tuned easily, without the risk of crashing.

The facility used is the Open Jet Facility of the TU Delft Aerospace Engineering faculty. It has a cross-section of 2.85m by 2.85m at the test area and can produce wind speeds of up to 30 m/s. The test setup can be seen in Figure 10. To simulate the fixed-wing aircraft, a cruciform beam is attached to the wind-tunnel roof, at the throat, allowing to maximize the approach path size. As can be seen in the image, the follower can be tethered to the roof with a rope and is operated by an assistant in order to always keep the rope slack, without allowing it to get tangled into the propellers. The facility contains a motion capture system called *Optitrack*, which can track position and attitude at mm accuracy and relay it to the UAVs via the ground station.

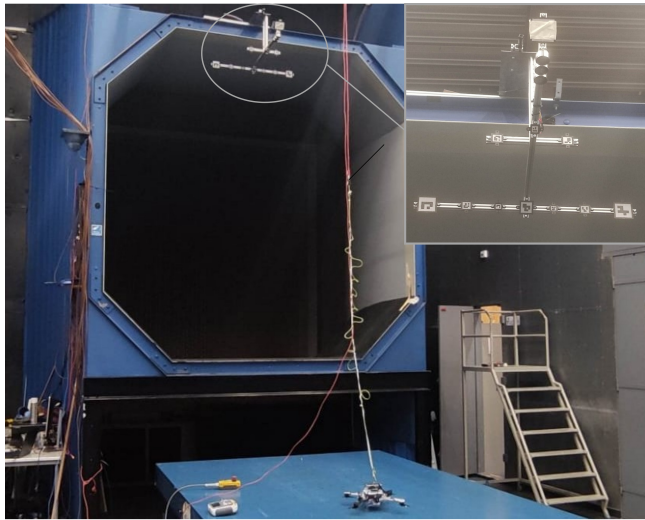


Figure 10: Wind tunnel test setup

4.1 Fixed-wing aircraft

The fixed-wing aircraft will be simulated by a cruciform beam and can be seen in Figure 10. Given that most aircraft are of fuselage + wing configuration, that was the shape used. The width and length are both 180mm. Thirteen Aruco markers were placed around the "airframe", varying in width, ranging from 2cm to 10cm. This enables the markers to be seen from further away, while still keeping full markers in view when in proximity. Three 25kg pot electromagnets are aligned to form the docking point. In future work, they will be spaced to ensure an exact docking position. The three electromagnets are powered with a 3-cell LiPO battery and can be turned off with a momentary-push button. The electromagnets can therefore be powered off to be able to simulate separation mid-flight.

4.2 Rotorcraft

The manufactured rotorcraft can be seen in Figure 2 and its hardware characteristics are tabulated in Table 1. This drone is an adaptation of an F350 drone frame, where the top and bottom plate are re-designed, and 3D printed to accommodate the three lateral motors. A longer landing gear was

also made to allow the lateral rotors to spin without hitting the ground. Due to the lateral motors, stronger vibration can now be expected around all three axes. Passive damping is therefore included, where the camera and the flight controller are both suspended on a damped rubber platform.

A flat metal plate is installed that will attach to the electromagnets. A thickness of 3 mm is chosen as it is a good compromise between pull force and mass. Finally, to avoid the tether from getting tangled into the back propeller, a simple propeller guard was included. Note that the four downfacing propellers are in pusher configuration, to assure the necessary clearance for docking.

Table 1: Follower platform characteristics

Take-off weight	1050 [g]
Frame:	Modified F350 plastic drone frame
Motors:	7 x Readytosky 2205 2700kV
ESCs:	Racestar V2 30A
Propellers:	GemFam 5146
Flight controller	Pixhawk 4
On_board computer	Raspberry-pi 4B
Camera	Raspberry-pi v2 noIR
Telemetry	SIK telemetry module
Flight software	Paparazzi

4.3 Experiment setup and results

Various tests were carried out to assess the feasibility and discover any challenges of the concept proposed. Three primary questions had to be answered. First, whether the platform can fly at forward velocities matching a fixed-wing aircraft while maintaining level and stable flight. Second, evaluate whether precise enough relative control can be achieved for rigid airborne docking to be feasible. And lastly, verify that the proposed docking method is viable and whether accidental detachments would not take place at these airspeeds.

To answer the first question, the aircraft was flown in the wind tunnel at a fixed position, while airspeed was incrementally increased up to 17.2m/s, at which speed the pusher propeller was saturated. The aircraft remained level with attitude deviations not surpassing 3° in all axes and its position within 5cm. The relative position tracking capabilities were not affected significantly when flying a flag in front of the aircraft at 14m/s.

The second question is answered by performing the approach and docking procedure in the wind tunnel, starting from the pre-docked position. The aircraft then autonomously approaches the target and docks onto the simulated fixed-wing aircraft. As soon as the operator sees that docking is successful, he turns off the motors and resets the desired position to the pre-docked setpoint. When ready, the operator arms the aircraft and pushes the button that disables the electromagnets. The over-actuated platform then returns to the pre-docked setpoint aggressively. This sequence can be seen

in Figure 11. The security tether is always left slack. Footage of this can be seen on Youtube ¹.

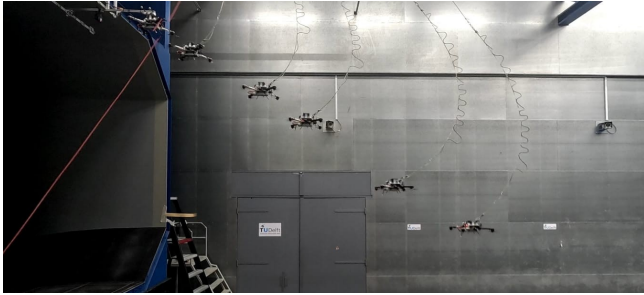


Figure 11: Approach sequence.

The approach path for 5 consecutive docking maneuvers at 13m/s can be seen in Figure 12. The rotorcraft uses Optitrack to reach the pre-docked setpoint, and from there, once the vision confidence is above the threshold, the docking procedure is started with the vision measurements. In these tests, the relative velocity is still obtained from Optitrack. Due to the markers being too small, the pre-docked setpoint had to be moved closer to -1.6m behind and 0.8m below the target. This problem can be mitigated by using larger markers, or increasing the resolution of the camera.

Due to the dimension constraints of the wind tunnel, the approach path is much more horizontal than intended. Because of this, a small deviation in height could lead to the rotorcraft striking the target at the end of the approach. To mitigate this, the final position setpoint is set to be 5cm below the target, and the rotorcraft is given an upward velocity command only if its horizontal position is within 5cm of the target. This resulted in a smooth, continuous docking sequence.

In the figure, the two dark dashed lines are the error threshold around the center line at any given position. This seems like a very narrow approach, but only the horizontal error to the current setpoint is taken into account for the advancement of the desired position setpoint. In fact, we see that the 5 approaches follow almost an identical path that is not through the center line. This is because there is no feed-forward control in place to make the rotorcraft exactly track the setpoint, and only the error controller makes the vehicle follow the setpoint, which is advancing faster than what the error controller can keep up with. The follower aircraft is effectively trailing the setpoint by the error threshold. This can be seen in Figure 12, where the rotorcraft (red dot) trails behind the setpoint (blue dot) by the threshold amount (red dashed lines). An animation is also available on Youtube¹. For this specific scenario, in which the leader is not moving and there is almost no turbulence, it makes the approach very smooth.

¹https://www.youtube.com/playlist?list=PL_KSX9GOn2P_KdCtqlExlus0Ogasmgbw

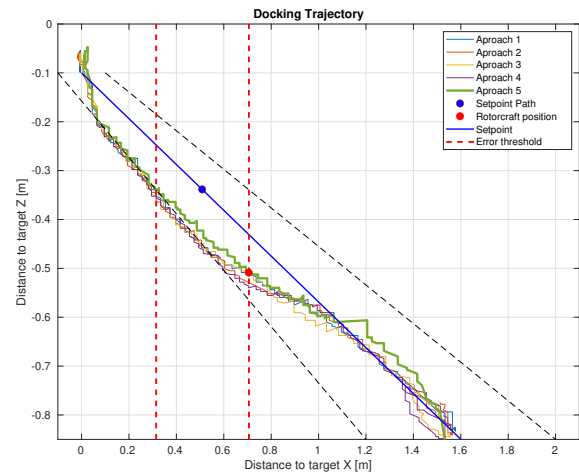


Figure 12: Path for 5 different approaches and setpoint position with respect to AC for the first approach.

4.4 Additional observations

While increasing the airspeed gradually from 0 to 15m/s , the aircraft starts oscillating heavily at a forward speed of 5m/s . The oscillations are mainly pitch oscillations, and it seems as if the back propellers are stalling and then recovering. This could be due to interaction between the front and back propellers. Even though the reason for this is still uncertain, it was easily mitigated by avoiding flying at that specific airspeed. In a mission for which this aircraft is designed, constantly flying at 5m/s is not a requirement.

5 CONCLUSIONS & RECOMMENDATIONS

In conclusion, this paper presents a proof of concept for rigid airborne docking using electromagnets between a fixed-wing aircraft and a hybrid rotorcraft. The successful demonstration of autonomous docking and undocking in a wind tunnel environment indicates the potential for future development and outdoor testing. The use of an INDI controller and horizontal motors for handling linear accelerations proved to be very effective and is recommended to be used in future work. The implemented cascaded proportional controller for position control, as well as the setpoint generation method used, yielded a smooth docking maneuver. The relative sensing method employing retroreflective ArUco markers and an infrared camera proved to work well in different lighting conditions. The methods presented in this paper should be tested in an outdoor environment with a real fixed-wing in future work. Overall, this study establishes the feasibility of rigid airborne docking and lays the groundwork for future advancements, holding promise for applications requiring precise docking between fixed-wing and rotorcraft vehicles.

REFERENCES

- [1] Nader S Labib, Matthias R Brust, Grégoire Danoy, and Pascal Bouvry. The rise of drones in internet of things: A survey on the evolution,

- prospects and challenges of unmanned aerial vehicles. *IEEE Access*, 9:115466–115487, 2021.
- [2] Alessandro Bacchini, Enrico Cestino, Benjamin Van Magill, and Dries Verstraete. Impact of lift propeller drag on the performance of evtol lift+ cruise aircraft. *Aerospace Science and Technology*, 109:106429, 2021.
 - [3] Adarsh Gangwal, Akshika Jain, and Shreeja Mohanta. Blood delivery by drones: A case study on zipline. *International Journal of Innovative Research in Science, Engineering and Technology*, 8(8), 2019.
 - [4] Hoodtech Mechanical. Flying launch and recovery system. hoodtech-mechanical. <https://www.hoodtechmechanical.com/vtol>. Accessed: 20-05-2023.
 - [5] PS Ramesh and JV Muruga Lal Jeyan. Comparative analysis of fixed-wing, rotary-wing and hybrid mini unmanned aircraft systems (uas) from the applications perspective. *INCAS Bulletin*, 14(1):137–151, 2022.
 - [6] Mads Friis Bornebusch and Tor Arne Johansen. Autonomous recovery of a fixed-wing uav using a line suspended between two multirotor uavs. *IEEE Transactions on Aerospace and Electronic Systems*, 57(1):90–104, 2020.
 - [7] Kristian Klausen, Jostein Borgen Moe, Jonathan Cornel van den Hoorn, Alojz Gomola, Thor I Fossen, and Tor Arne Johansen. Coordinated control concept for recovery of a fixed-wing uav on a ship using a net carried by multirotor uavs. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 964–973. IEEE, 2016.
 - [8] Kristian Klausen, Thor I Fossen, and Tor Arne Johansen. Autonomous recovery of a fixed-wing uav using a net suspended by two multirotor uavs. *Journal of Field Robotics*, 35(5):717–731, 2018.
 - [9] Peter R Thomas, Ujjar Bhandari, Steve Bullock, Thomas S Richardson, and Jonathan L Du Bois. Advances in air to air refuelling. *Progress in Aerospace sciences*, 71:14–35, 2014.
 - [10] Daniel Briggs Wilson. *Guidance, Navigation and Control for UAV Close Formation Flight and Airborne Docking*. PhD thesis, University of Sydney, 2015.
 - [11] DARPA. Gremlins program demonstrates airborne recovery. Defense Projects, 05 2021.
 - [12] Inyeni Amakuro Showers et al. *Development of an autonomous airborne docking and undocking system for micro drones to a mother drone*. PhD thesis, Cape Peninsula University of Technology, 2019.
 - [13] Kevin Ling. Precision landing of a quadrotor uav on a moving target using low-cost sensors. Master's thesis, University of Waterloo, 2014.
 - [14] Alexandre Borowczyk, Duc-Tien Nguyen, André Phu-Van Nguyen, Dang Quang Nguyen, David Saussié, and Jerome Le Ny. Autonomous landing of a multirotor micro air vehicle on a high velocity ground vehicle. *Ifac-Papersonline*, 50(1):10488–10494, 2017.
 - [15] 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.
 - [16] 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.
 - [17] Muhammad Tahir, Sayed Saad Afzal, Muhammad Saad Chughtai, and Khurram Ali. On the accuracy of inter-vehicular range measurements using gnss observables in a cooperative framework. *IEEE Transactions on Intelligent Transportation Systems*, 20(2):682–691, 2018.
 - [18] Adam Marut, Konrad Wojtowicz, and Krzysztof Falkowski. Aruco markers pose estimation in uav landing aid system. In *5th International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, pages 261–266. IEEE, 2019.

Task Allocation for Heterogeneous Unmanned Aerial Vehicles

A. Schwager*, P. Müller, F. Knaak, and D. Moormann

Institute of Flight System Dynamics at RWTH Aachen University, Willnerstraße 7, 52056 Aachen

ABSTRACT

To reduce the workload of operators in situations like coordinated search and rescue missions, in which a large number of heterogeneous Unmanned Aerial Vehicles (UAVs), are used, an automated fleet management can be of significant benefit. As part of the fleet management the different tasks must be allocated to the available UAVs. Multiple algorithms have been developed in the past to perform task allocation without any conflicts. In this paper we provide a concise introduction to prevalent algorithms and adapt a task allocation algorithm to work in dynamic environments, such as those commonly found in rescue operations. Based on the Asynchronous Consensus-Based Bundle Algorithm, the new “central mediator” is introduced, which guarantees a conflict-free allocation at every step of the allocation process and eliminates the communication between the agents representing the UAVs. The presented algorithm is applicable in any environment with reliable network connections between the “central mediator” and the agents. Furthermore, the algorithm is designed to be part of a fleet management system that considers heterogeneous UAVs.

1 INTRODUCTION

Unmanned Aerial Vehicles (UAVs) are becoming more vital to the efforts of emergency responders due to their cost and resource efficiency and lower operational risk compared to conventional means of aerial or ground reconnaissance. This is especially true in the case of major incidents, where nowadays an increasing number of different rescue organisations involved in the disaster management is using UAVs. This leads to a wide variety of UAVs present at the site, each suited for their dedicated purpose.

The resulting heterogeneous fleet of UAVs will possess a broad range of operational airspeeds and ranges. Moreover, in a highly dynamic environment, emergency responders must coordinate the said fleet of UAVs. Simultaneously the airspace in the vicinity of the incident is shared with other

airborne rescue forces including rescue helicopters. All of these factors would lead to a substantial workload for the fleet operator, if no automation is implemented.

This paper is based on the requirements of the GrenzFlug+ research project, which focuses on the use of UAVs in cross-border rescue operations. The project aims to automate the fleet and task management to reduce the workload of rescue forces. Objectives and their order of priority are defined by the rescue forces. These objectives are subsequently transformed into tasks by the “Task Management”, as shown in Figure 1, and assigned a priority accordingly. In the subsequent stage the UAVs are assigned these tasks. This paper focuses on the second step, “Task Allocation”, and assumes that the “Task Management” has provided the corresponding tasks and priorities. The “Fleet Management” reduces the operator’s workload by eliminating the need to consider UAV-specific parameters when defining objectives.

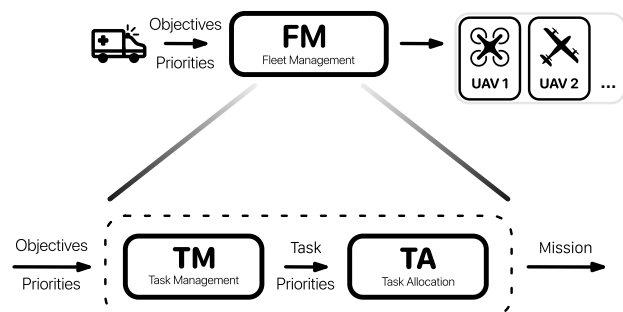


Figure 1: Generalized System Overview

To address the task allocation, a distributed algorithm based on the Asynchronous Consensus Based Bundle Algorithm (ACBBA) [1] is developed. In the algorithm each UAV is represented by an agent, which results in an agent-based approach.

In the opening section of this paper we will provide a brief overview of related work. Next, we explain the proposed approach for the task allocation algorithm, outlining the architecture, introducing the “central mediator” and giving a straight forward example. The paper concludes with the presentation of the analysis of the results of the implemented

*Email address(es): schwager@fsd.rwth-aachen.de

algorithm for some test cases.

2 RELATED WORKS

2.1 Management in Multi-UAV environments

There are different architectures for the management of numerous heterogeneous UAVs. To enhance comprehension of the various architectures, Figure 2 provides an overview of the different topologies.

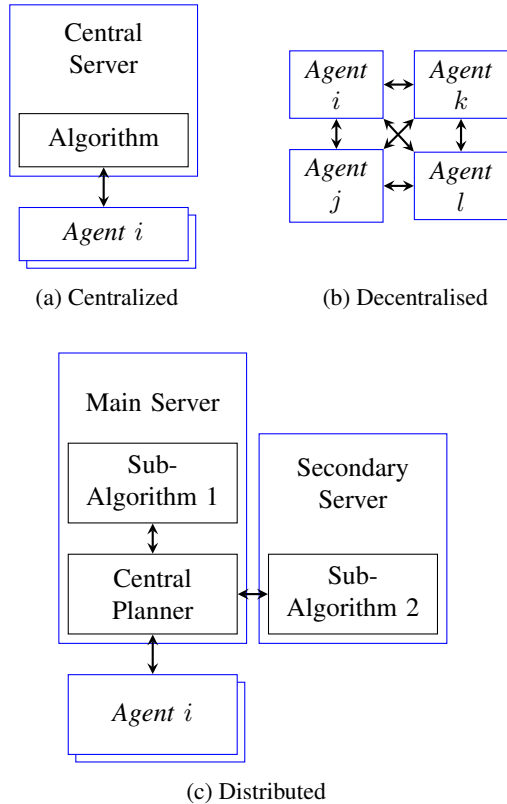


Figure 2: Overview of different architectures

In a centralised architecture (Figure 2 (a)) the entire task allocation problem is solved as a single optimisation problem. The advantage of centralised architectures is that they rely on a reduced amount of communication needed during the computation, as the algorithms gather relevant information a-priori and can be executed on modern multi-core computing systems. The communication performance of the single computing machine may be a limiting factor, as it needs to receive all the data relevant to fleet task management, including the necessary data from the UAVs, services supplying airspace data and other relevant data sources. In sparsely connected environments the problem may occur that the central solver may find a solution, but cannot transmit it to all agents for execution. This may result in conflicts as not all agents work with the same task allocation. [2]

In distributed architectures (Figure 2 (c)), there are separate computing modules communicating with each other. Depending on whether they are implemented on a single machine or on separate machines, communication can be over shared memory or network connections. Distributed architectures lose performance compared to a centralised architecture as additional communication is required but enable the system to exploit the computing power of multiple machines. In contrast to decentralised architectures, the distributed architecture relies on high performance and reliable communication between the different computing machines. [2]

A decentralised architecture (Figure 2 (b)) omits the central computing unit and performs all computation on-board of the UAVs. As each agent solves the task allocation problem separately, communication is only needed to negotiate between the agents. This has the advantage, especially for sparsely connected systems, that the agents can find a solution with minimal input from other systems. Decentralised architectures are usually not designed to find the mathematically optimal solution but a solution close to the optimum to reduce the required computation time. The disadvantage is the often reduced computing performance when operating in a robust network. [2]

As discussed in Section 3.1, a distributed environment with a decentralised algorithm is chosen because it allows a flexible architecture while reducing the necessary communication with the UAV. A brief overview of the algorithms used is given below.

2.2 Consensus-Based Bundle Algorithm (CBBA)[3]

The Consensus-Based Bundle Algorithm (CBBA) is an algorithm developed using a decentralised architecture. It uses a bidding process to allocate tasks to agents. The algorithm consists of two phases:

First, in the bundle construction phase, each agent creates a bundle of tasks it wants to perform. After all agents have completed the first phase, the consensus phase starts. In this phase the agents need to exchange information about the bundles. With the information from the other agents each agent now checks whether it has been outbid by other agents for the same task. In case it has been outbid, it cannot carry out that task. As the planning for all the tasks after this one in the bundle has also become invalid, the agent must release the rest of the bundle along with the task or which it was outbid. With the reduced bundles, the bundle generation phase starts again.[4]

Choi et.al. [4] have shown that the CBBA ensures 50% optimality. As per an empirical investigation, one may anticipate performance within 93% of the optimal solution in practice [5].

2.3 Asynchronous Consensus-Based Bundle Algorithm[1]

The ACBBA is a modification of the CBBA that includes support for asynchronous operations. Similar to the CBBA, the ACBBA is designed for execution in a decentralised environment. However while the CBBA necessitates waiting for all agents to complete the bundle generation before beginning the second phase. The ACBBA modifies the consensus phase to enable bundle generation phase and consensus phases to be executed asynchronously across various agents. During the modified consensus phase, five distinct actions can be taken upon receiving a new plan from another agent.

1. Update & Rebroadcast: Update the own state and broadcast the received plan
2. Leave & Rebroadcast: Leave own state and broadcast it instead of the received plan
3. Leave & No-Rebroadcast: The own plan is not changed and nothing is broadcasted
4. Reset & Rebroadcast: The own plan is deleted and the received plan is broadcasted
5. Update Time & Rebroadcast: Update the time stamp and broadcast the own plan with the update's time stamp

Actions 2, 4 and 5 are needed to resolve conflicts or inconsistencies between the current states of the various agents. [1]

The ACBBA has been shown to be convergent in dynamic environments as long as it can reach a solution before the environment is modified by changing the tasks to be allocated. For dynamic environments where tasks are added and deleted quicker than convergence is reached the algorithm remains operational but does not reach a convergent state. Due to the nature of the ACBBA, an analytical evaluation of the optimality of the algorithm is not possible. [1]

For lossy networks, it has been shown that the ACBBA may not always find a conflict-free solution. In order to achieve such a solution, additional measures that increase the negotiation overhead are needed [6]. This paper addresses this issue by implementing the algorithm in a distributed environment and introducing the "central mediator".

3 CONCEPT

3.1 Architecture

In a decentralised architecture, it would be necessary to compute all paths and their corresponding costs on-board. These calculations would be performed on the same machine as the flight safety critical processes, such as the conflict avoidance, or would require additional hardware, which would decrease flight performance. The sharing of these resources is only acceptable if the path planning is performed infrequently. For the given use-case the task distribution and

segmentation shall be optimized for the available UAVs. This requires more iterations and an increased computational effort on each UAV. To meet this requirement, a distributed architecture was chosen for the given use-case. The algorithm runs on a single high-performance machine to ensure the necessary computational effort. For each UAV, the architecture features an individual agent that runs on a separate thread on the same machine as the central coordinating instance.

A further advantage of employing a distributed architecture, as opposed to a decentralised one, is that the UAVs do not require constant communication with one another and that no external data, such as environmental information, must be transferred to them. This reduces the bandwidth required for mission planning and therefore frees up bandwidth for communication relevant to flight safety.

As the fleet management system incorporates heterogeneous UAVs, it is feasible that tailored path planning tools for individual types of UAV may be utilised in the future. This would prove a challenge in a centralised architecture. In addition, employing a distributed approach enables future expansion of the fleet task management system via inclusion of new modules.

3.2 Task Allocation

For the task allocation within the fleet task management the ACBBA [1] has been selected as the foundation. The ACBBA is preferred due to its ability to support asynchronous operations and independence from the underlying path planner. This is a necessity when using different path planners with different execution times for a heterogeneous fleet. Furthermore, the ACBBA can adapt to a dynamic environment where new tasks are constantly added and removed; a critical feature when the task allocation algorithm is employed with the task management.

In the ACBBA all calculations are performed by the agent's on-board processing unit. In a first adaption step all calculations are moved into the distributed environment, where separate threads on the same machine perform calculations for each UAV. This allows the ACBBA's asynchronous operations to be maintained while taking advantage of the benefits of a distributed architecture.

As the ACBBA was developed for a decentralised environment, it requires significant effort to resolve conflicts arising from each UAV's individual current bidding result [1][6]. However, in a distributed environment where communication is assumed to be nearly instant and fully reliable, these complex algorithms can be replaced by a central negotiation instance - the mediator. The bundle generation phase defined in the CBBA [4] is carried out independently and asynchronously for each agent, with the consensus phase shifted from the agents to the mediator. By achieving conflict-free resolution following each iteration, the task can be terminated

and the current solution can be utilised without necessitating additional conflict checks.

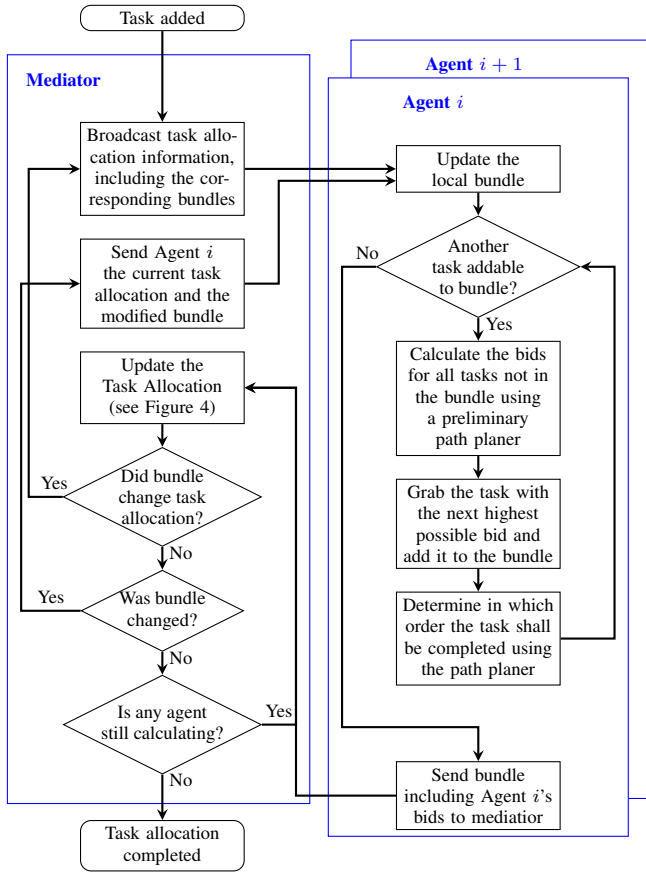


Figure 3: Flow Chart of the Task Allocation Algorithm

The procedure utilized in the proposed adaptation of the ACBBA is illustrated in Figure 3. The initial step involves broadcasting the existing tasks, along with the corresponding winning bid and the currently winning agent for each task, to all agents. In the first broadcast no winning bids and agents are included as there has been no bidding process yet. With the task list and the current allocation each agent attempts to secure additional tasks by bidding higher than the currently allocated agent.

Firstly the agent calculates its bid for all tasks not yet in its own bundle. Next, the agent determines whether its bid would surpass the current winning bid. Finally, among all the bids that beat the current winning bid, the agent selects the one with the highest bid. Afterwards, the path planner is executed to generate a path that considers all tasks in the updated bundle, including the newly added task. These steps are repeated until the agent is unable to add further tasks to its bundle. This happens either when the agent has added all the tasks where it can outbid the current winning agent or, when it is unable to complete any more tasks due to

reaching its limitations, which vary depending on the UAV represented by the agent. A possible example for such a limit is that the maximum flight distance achievable by the UAV has been reached or that the remaining tasks are unreachable due to obstacles.

For the specific application of a major disaster, the key consideration is the timeline for completing the task and the task's priority. Task priority measures the value of a task and is defined by the "Task Management". The time taken to complete a task is calculated by the agent and represents the cost. The bid is determined by employing equation (1) in which j denotes the task number and a_i the agent currently performing the bundle generation. The resulting bid, $c_{a_i,j}$ is determined by the task priority p_j , the time required for task completion $t_{completion,j}$ and the scaling factor T . For the scope of this paper $T = 1$ s will be assumed.

$$c_{a_i,j} = p_j - \frac{t_{completion,j}}{T} \quad (1)$$

Once the Agent i has completed the bundle generation, it transfers the bundle along with the corresponding bids to the mediator. As the mediator always stores the most recent task allocation, the assessment of the update times for each of the bids can be omitted, thus simplifying the consensus phase. During the mediation phase, the mediator iterates through the bundle of Agent i , as depicted in Figure 4. It should be noted that a consensus phase may lead to a task, previously allocated to an agent, having no agent assigned.

If, during the consensus phase, there is a change in the task allocation, the new allocation is broadcasted to all agents along with their updated bundle, thus initiating a new bundle generation phase. If the task allocation remains unchanged, but Agent i 's bundle changes, only Agent i will be send the updated information as only it needs to repeat the bundle generation phase. If neither task allocation nor bundle are changed, Agent i has achieved consensus with the present task allocation. If no other agent is performing a bundle generation phase at this point, the final task allocation was found.

3.3 Simplified Example Scenario for the Task Allocation Algorithm

For a improved comprehension of the adapted algorithm an example involving three tasks and two UAVs is given (see Figure 5). The Agent TW (a_{TW}) is a tilt-wing aircraft flying in fixed-wing configuration with a constant ground speed of $20 \frac{m}{s}$, whereas the Agent MR (a_{MR}) is a multi rotor system flying with $10 \frac{m}{s}$ ground speed. All three tasks are waypoints to be reached with a priority factor $p_j = 20$. For the sake of simplicity, curves are omitted.

Each agent has a bundle vector storing the current bundle ($b_{a_{TW}}$ and $b_{a_{MR}}$) and a corresponding bid vector ($c_{a_{TW}}$ and $c_{a_{MR}}$). As a first step the mediator broadcasts the complete list of all tasks, with no winning agent and no winning bids.

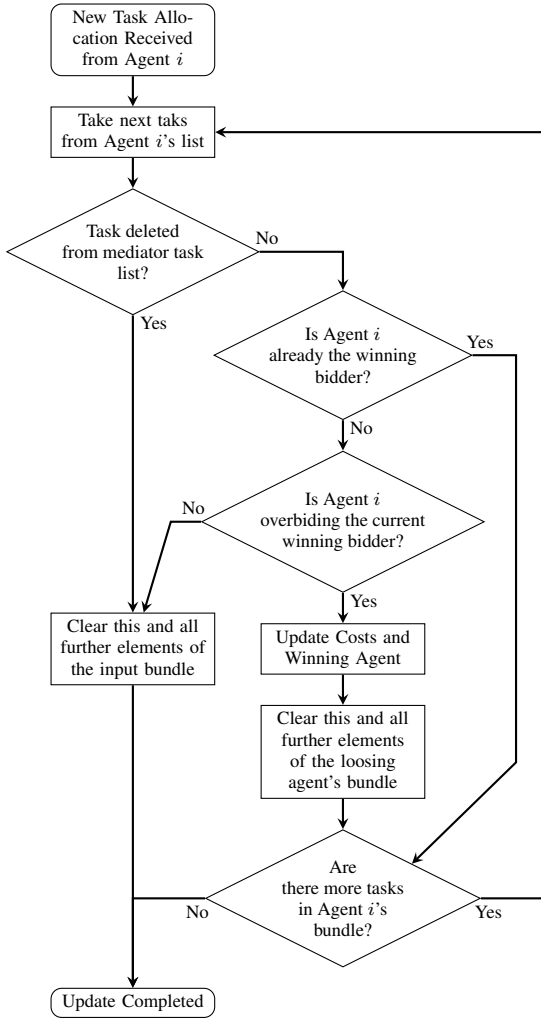


Figure 4: Flow Chart of the consensus step performed by the mediator

From the given allocation both agents generate their corresponding bundles. For the given case Agent 1 generates the bundle $b_{a_{TW}}^I = [t_1 \ t_2 \ t_3]^T$ with the corresponding bids $c_{a_{TW}}^I = [18 \ 15 \ 12]^T$. Agent TW is quicker in completing the bundle generation phase and therefore initiates the first mediation phase. After this mediation phase the mediator has the task allocation shown in Table 1.

	$a_{winning}$	$c_{winning}$
t_1^I	a_{TW}	18
t_2^I	a_{TW}	15
t_3^I	a_{TW}	12

Table 1: Task Allocation after the first consensus phase with the bundle from Agent TW

As the task allocation has changed, the mediator broad-

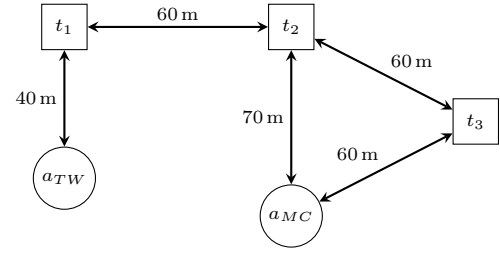


Figure 5: Locations of tasks and UAVs for the simplified example

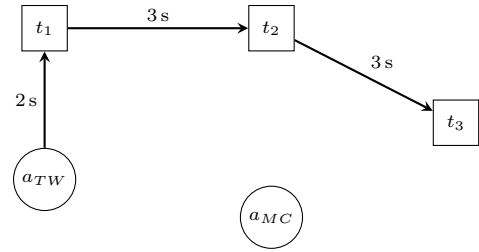


Figure 6: Simplified path planned by a_{TW}

casts it to both agents. As Agent TW determines that the allocation corresponds with its own bundle and thus, does not perform a new bundle generation phase while Agent MR is still running its first bundle generation phase. Therefore this broadcast does not trigger any new bundle generation phases at this moment. When Agent MR completes its initial bundle generation phase, it triggers the second mediation phase with the bundle $b_{a_{MR}}^I = [t_3 \ t_2 \ t_1]^T$ and the corresponding bids $c_{a_{MR}}^I = [14 \ 8 \ 2]^T$.

	$a_{winning}$	$c_{winning}$
t_1^I	a_{TW}	18
t_2^I	a_{TW}	15
t_3^I	a_{MR}	14

Table 2: Task Allocation after the second consensus phase with the bundle from Agent MR

As the Agent MR has out-bid Agent TW on on the first

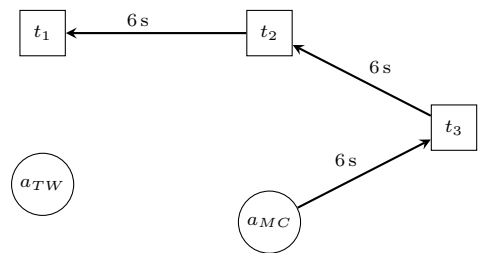


Figure 7: Simplified path planned by a_{MR}

task in its bundle and has been out-bid on the second task in its bundle, the task allocation and both agents will perform a new bundle generation phase using the input data from Table 2. During this bundle generation phase Agent *MR* generates the bundle $b_{a_{MR}}^{II} = [t_3]$ and the corresponding bids $c_{a_{MR}}^{II} = [14]$. Agent *MR* does not select tasks t_1 and t_2 as it is aware that it cannot outbid Agent *TW* for these tasks. Similarly, Agent *TW* does not select task t_3 as it acknowledges that it cannot outbid Agent *MR* for this particular task. Since neither of the agents modified its bundle, the allocation remains unchanged in the following mediation phases. Thus, the task allocation algorithm has reached a convergent state.

4 RESULTS

The task allocation algorithm, as outlined in section 3.2, is implemented using ROS2 with C++ and Python being the primary languages employed. This implementation was tested with varying number of tasks, each requiring a set point to be reached under a specific heading. The test scenario was tackled by 10 agents, each planning dubbins paths to complete the different tasks. The number of agents was selected to compare the outcomes of the ACBBA presented in [1] and the results attained through the algorithm proposed within this paper.

With the described scenario and path planner it was found that the bundle generation phase was at least four times slower than the mediation phase for the lowest number of tasks, while it was approximately fifteen times slower for the largest number of tasks. As 10 agents were used to tackle the given tasks the mediator becomes a bottleneck when the bundle generation phase is less than 10 times slower than the mediation phase. Since the utilized path planner does not consider obstacles or flight performance, it may be assumed that this issue will be resolved with the implementation of more advanced path planners.

The scenarios were executed on various systems, enabling the agents to compute independently. During the evaluation of the results, it is evident that the number of mediations is directly proportional to the time required to reach a convergent state for a given system. To have a measure of the convergence speed that is independent from the processor speed the number of mediations was used to evaluate the results. Figure 8 illustrates the number of messages against the number of tasks in the specified scenario.

For the test scenarios, a linear relationship is evident between the number of tasks and the mediator steps required. This is the expected behaviour as an increase in number of tasks leads to a larger bundle size, resulting in a more dropped tasks during the mediation phases when an agent is out bidden by another agent.

For the comparison between the ACBBA [1] and the developed algorithm, it is necessary to compare the number of

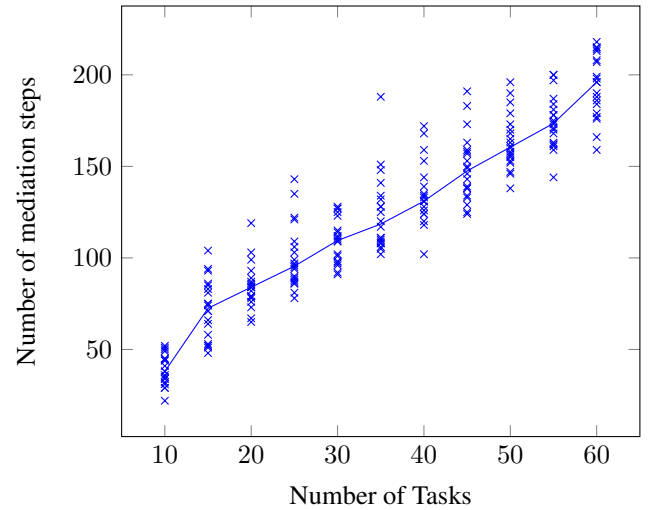


Figure 8: Number of mediator steps required to reach consensus for the given number of tasks including the median

messages used by the ACBBA and the number of mediation phases required. However, it is important to note that one mediation phase is equivalent to $n_{msg} \approx n_{agents} - 1$ messages. This relation results from broadcasting of the task allocation at the end of the mediation phase, which is the equivalent to sending a message to each of the nine other agents.

When comparing the performance of the ACBBA and the developed mediator-based algorithm, three sections exhibit distinct comparison results. For the scenarios with 20 or fewer tasks the performance of both algorithms is similar with a slight advantage for the developed mediator approach. For the scenarios with $20 < n_{task} \leq 35$, the developed mediator-based algorithm proves significant superior to ACBBA. For instance, in the scenarios with 30 tasks, the median number of messages required by the ACBBA is thrice that of the developed mediator-based algorithm.

For the scenarios featuring over 35 tasks, the ACBBA performs increasingly better, so that the median number of messages required by ACBBA for the scenarios with 60 tasks is only 30% higher than that of the developed mediator-based algorithm. An explanation for the reduction of required messages was not provided by Johnson et.al. [1]. However, it can be assumed that the agents have a limit on the maximum path length. This would result in fewer conflicts to resolve for higher numbers of tasks, thus reducing the number of messages required. The simulation of the developed mediator-based algorithm did not have a similar limit. Therefore, the comparability of the two algorithms is reduced for all scenarios with more than 35 tasks.

5 CONCLUSION

An adapted task allocation algorithm is presented for the use case of a connected UAVs in a search and rescue scenario. The algorithm has been designed to run in a

distributed environment and is based on the Asynchronous Consensual Bundle Based Algorithm [1]. This choice was made because the algorithm enables the integration of heterogeneous UAV as agents. To improve performance the consensus phase of ACBBA has been replaced by the newly introduced mediation phase, which is executed by the new central mediator. The performance improvement was demonstrated performing simulation runs of the developed mediator-based algorithm and comparing the results with simulation runs for the original ACBBA.

The task allocation algorithm has been designed with future extendability in mind. A planned extension is integrating an interface to the U-space system for the planning phase, in order to identify potential conflicts and request prioritisation or perform strategic deconfliction.

Further work is needed to determine if the mediator-based algorithm developed can also exhibit a reduction in required messages, as demonstrated by the ACBBA, for a larger number of tasks. . The function used to compute each agent's bid is also a topic for future work, as is the impact of the introduction of a range-limit for the agents.

ACKNOWLEDGEMENTS

The project "GrenzFlug+" is funded by the German Federal Ministry of Digital Affairs and Transport as part of the "Innovative Air Mobility" funding guideline based on a resolution of the German Bundestag.

REFERENCES

- [1] Luke Johnson, Sameera Ponda, Han-Lim Choi, and Jonathan How. Asynchronous decentralized task allocation for dynamic environments. In *Infotech@Aerospace 2011*, volume 25, pages 912–926. American Institute of Aeronautics and Astronautics, March 2011.
- [2] Sameera S. Ponda, Luke B. Johnson, Alborz Geramifard, and Jonathan P. How. Cooperative Mission Planning for Multi-UAV Teams. In Kimon P. Valavanis and George J. Vachtsevanos, editors, *Handbook of Unmanned Aerial Vehicles*, pages 1447–1490. Springer Netherlands, Dordrecht, 2015.
- [3] Luc Brunet, Han-Lim Choi, and Jonathan How. Consensus-Based Auction Approaches for Decentralized Task Assignment. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, Hawaii, August 2008. American Institute of Aeronautics and Astronautics.
- [4] Han-Lim Choi, L. Brunet, and J.P. How. Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics*, 25(4):912–926, August 2009.
- [5] Luca Bertuccelli, Han-Lim Choi, Peter Cho, and Jonathan How. Real-Time Multi-UAV Task Assignment in Dynamic and Uncertain Environments. In *AIAA Guidance, Navigation, and Control Conference*, Chicago, Illinois, August 2009. American Institute of Aeronautics and Astronautics.
- [6] Matthew Rantanen, Nicholas Mastronarde, Jeffrey Hurdack, and Karthik Dantu. Decentralized Task Allocation in Lossy Networks: A Simulation Study. In *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9, Boston, MA, USA, June 2019. IEEE.

Time-Optimal Trajectory Generation with Input Boundaries and Dynamic Waypoints Constraints for A Swarm of Quadrotors

Yuyang Shen¹, Fangguo Zhao^{1,2}, Jiahao Mei³, Jin Zhou¹, Jinming Xu¹ and Shuo Li¹ *

¹. Zhejiang University, China

². Northwestern Polytechnical University, China

³. Zhejiang University of Technology, China

ABSTRACT

Challenges in aggressive flights for a swarm of quadrotors lie in optimal trajectory generation technologies. However, using the quadrotors' physical input limits as the input boundaries in the optimization problem leaves no control margin to handle the deviation caused by disturbance, model inaccuracy and sensor imperfection, etc. Artificially setting input boundaries cannot guarantee any optimality. In this article, we describe a novel optimal trajectory generation method considering flying time and input boundaries for a swarm of quadrotors to fly through pre-defined waypoints including dynamic waypoints without collisions. We verify the method in the Gazebo simulations where a swarm of 5 quadrotors can fly through a complex 6-waypoint track including 1 dynamic waypoint in a $35m \times 35m$ space. Flight tests are performed on two quadrotors passing through 3 waypoints in a $4m \times 2m$ flight arena to demonstrate the feasibility of the proposed method in the real world.

1 INTRODUCTION

Autonomous quadrotors are widely applied and play an important role in multiple application scenarios and show their potential applications such as disaster response and air delivery scenarios. Due to their inherent agility, autonomous aggressive flight has become a very hot research area in the robotics community in recent years.

The most classic aggressive trajectory generation method is the differential-flatness based methods [1, 2, 3], which connect the waypoints using polynomials while keeping the quadrotor's inputs within pre-defined boundaries. These methods are very straightforward and can be easily deployed on real quadrotors. Thus, they are widely used in aggressive flight applications. However, although the traveling time between two waypoints can be minimized, the resulting polynomials make the trajectories far from the true optimal as the

input curves cannot stay on the input boundaries for a long time and usually, the input curve just touches the boundary and then is bounced back immediately.

Another category of methods of generating aggressive trajectories for quadrotors are optimization-based methods. These methods treat the quadrotor's states and inputs as optimization variables and the quadrotor's dynamics, initial states, target states, waypoints, etc as constraints to formulate an optimization problem. By solving the optimization problem, the optimal inputs and the corresponding states can be calculated to minimize an optimization object such as flying time, total energy or a combination of both. For example, in [4], the time-optimal trajectories are generated that can guide one quadrotor to fly through pre-defined waypoints and the quadrotor's speed has surpassed the human pilots. To make the optimal guidance and control method onboard, machine learning techniques are used to learn the off-board generated optimal trajectory libraries [5, 6, 7]. However, these methods don't take the quadrotor's input boundaries into consideration in their time optimization problems. If the input boundaries are set exactly the same as the quadrotor's physical limits, the resulting trajectories should be very close to the 'theoretical' optimal trajectories. However, in real-world flights, the quadrotor will have no more ability to handle the deviation caused by disturbance, model inaccuracy and sensor imperfection, etc. A common way to handle this is to artificially lower the inputs' boundaries to leave some capacity for the feedback controllers to handle the deviations. However, artificially setting input boundaries cannot guarantee any optimality. Furthermore, most research about the quadrotor's aggressive flight focuses on single drone scenarios instead of a swarm of quadrotors.

In term of quadrotors' swarm, the majority of research use polynomials to navigate quadrotors but the optimality cannot be guaranteed [8, 9]. In [10], a swarm of micro quadrotors were used for the search and rescue tasks in an unknown environment. Carlos et al. presented a novel real-time and multi-vehicle motion planning framework, enabling complex transition tasks to be performed [11]. Duisterhof et al. used a swarm of quadrotors to seek gas-leaking in cluttered environments [12]. Zhou et al. realized a swarm of quadrotors flying through a bamboo forest [13]. However, the velocity of

*Email address(es): shuo.li@zju.edu.cn

the quadrotors in above research is still far from the quadrotors' limits. In our previous work [14], we proposed a novel optimization-based trajectory generation method for a swarm of racing drones that can generate the time-optimal trajectories for quadrotors to fly through the pre-defined waypoints while avoiding collisions with each other. In this paper, based on our previous work, we

1. jointly optimize the flight time and input boundaries in a newly defined optimization objective to leave control margins to handle disturbance while keeping high speed
2. add dynamic waypoint constraints to the optimization problems, which enables a swarm of quadrotors to fly through a more complex track
3. valid and analyze the proposed method in simulations where 5 quadrotors can fly through 6 waypoints with a maximum speed of $14m/s$ and also in the real-world flight tests to demonstrate the feasibility of the proposed method.

2 PRELIMINARY

For the benefit of the readers, we present the quadrotor's dynamics model below,

$$\dot{\mathbf{x}} = \mathbf{f}_{dyn}(\mathbf{x}, \mathbf{u}) = \begin{cases} \mathbf{v} \\ \mathbf{g} + \frac{1}{m} \mathbf{R}(\mathbf{q}) \mathbf{T} \\ \frac{1}{2} \Lambda(\mathbf{q}) \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \\ \mathbf{J}^{-1}(\boldsymbol{\tau} - \boldsymbol{\omega} \times \mathbf{J} \boldsymbol{\omega}) \end{cases} \quad (1)$$

where

$$\mathbf{T} = \begin{bmatrix} 0 \\ 0 \\ \sum T_s \end{bmatrix}$$

is the thrust vector of the quadrotor and

$$\boldsymbol{\tau} = \begin{bmatrix} l/\sqrt{2}(T_1 + T_2 - T_3 - T_4) \\ l/\sqrt{2}(-T_1 + T_2 + T_3 - T_4) \\ c_\tau(T_1 - T_2 + T_3 - T_4) \end{bmatrix}$$

is the torque vector of the quadrotor. In the equations above, $\mathbf{v}$, $\mathbf{q}$ and $\boldsymbol{\omega}$ are velocity, quaternions and angular velocity of the quadrotor, respectively. $\mathbf{R}(\mathbf{q})$ is the rotation matrix. l is the arm length. T_s is the thrust provided by the s^{th} rotor, which are the inputs of the dynamic system (1).

In the original CPC method [4], a progress measure variable matrix $\boldsymbol{\lambda}$ and a progress change matrix $\boldsymbol{\mu}$ are added to the optimization problem to control the assignment of the waypoints to ensure that one quadrotor can fly through the pre-defined waypoints. To be more specific, the trajectories

are discretized to N nodes while matrix $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$, which are the optimization variables, are used to assign the nodes to the waypoints. The optimization goal is to adjust the flight time t_f (also the time interval between two nodes $\Delta t = t_f/N$) of the entire trajectory while satisfying the waypoint pass-through constraints and the quadrotors' dynamics constraints. However, the CPC method cannot be directly extended to multi-drone scenarios as each quadrotor has its own t_f so the nodes cannot be aligned. As a result, it is difficult to add collision constraints to the original optimization problem.

Thus, in our previous work [14], a novel method was proposed to generate time-optimal trajectories for a swarm of autonomous racing drones so that during the flight they can fly through the pre-defined waypoints while avoiding each other and arrive at the goals with minimum time. To address collision avoidance between the quadrotors, the nodes of the quadrotors are synchronized by using the fixed time interval Δt . Then, the optimization problem is transformed from adjusting the total flight time t_f while the quadrotor arrives at the goal at the last node to adjusting the arrival node with the fixed time interval Δt . The constraints are listed in equations below. Here, we use a left subscript i to indicate that the variables belong to the i^{th} quadrotor.

$${}_i \lambda_k^j \leq {}_i \lambda_k^{j+1} \quad (2)$$

$${}_i \lambda_{k+1}^j - {}_i \lambda_k^j + {}_i \mu_k^j = 0 \quad (3)$$

$${}_i \mu_k^j (\|{}_i \mathbf{P}_k - {}_i \mathbf{P}^{wj}\|_2^2 - {}_i \nu_k^j) := 0 \quad (4)$$

$$\|\mathbf{E}({}_i \mathbf{P}_k - {}_r \mathbf{P}_k)\|_2^2 - \delta_{col} \geq 0 \quad i \neq r \quad (5)$$

where ${}_i \lambda_k^j$ is a bool variable indicating if the i^{th} quadrotor has flew through the j^{th} waypoint at time t_k . ${}_i \mu_k^j$ means if the i^{th} quadrotor is passing through the the j^{th} waypoint. ${}_i \nu_k^j$ is a tolerance slack. The operator ' $:=$ ' means a NAND (not and) function. ${}_i \mathbf{P}_k$ is the position of the i^{th} quadrotor at time t_k . ${}_j \mathbf{P}^{wp}$ is the position of the j^{th} waypoint and $\delta_{col} > 0$ is the tolerance ensuring that two quadrotors do not collide with each other. $\mathbf{E}$ is the matrix used to relieve downwash risk, making quadrotors try to avoid collisions in x and y directions instead of z direction. The readers are referred to [14] for details of the proposed method.

As stated before, if we set the input boundaries of the optimization problem to be the real quadrotor's physical limits, the quadrotor will have no ability to handle the deviation caused by disturbance, model inaccuracy, sensors' imperfection, etc, which in turn can significantly affect the flight

performance or even cause failures. A common way to handle this is to artificially lower the inputs' boundaries to leave some capacity for the feedback controllers to handle the deviations. However, artificially setting input boundaries cannot guarantee any optimality. Thus, in the following part, we model the flying time and input boundaries in our trajectory optimization problem to generate trajectories that can guide a swarm of quadrotors to fly through the pre-defined waypoints while achieving the minimal optimization objective.

3 METHODOLOGY

3.1 Dynamic Waypoint Constraints

Inspired by some aggressive flight scenarios such as autonomous drone racing where waypoints/racing gates are not only static but also moving with some specific patterns [15, 16], in this article, one of the waypoints moves with a pre-defined trajectory while others keep static during the flight. Thus, the generated trajectories may guide different quadrotors to fly through the dynamic waypoint at different locations. In our previous work [14], the corresponding constraints are expressed in equation (4).

However, in dynamic-waypoint scenarios, the position of the waypoints is no longer a constant and it becomes a function of time. Thus, constraints (4) becomes

$${}_i\mu_k^j(\|{}_i\mathbf{P}_k - {}_i\mathbf{P}(k \cdot \Delta t)^{wj}\|_2 - {}_i\nu_k^j) := 0 \quad t \in [0, N], \quad (6)$$

where t is the k^{th} node.

3.2 Input Boundary Constraints

In the previous work, the optimization objective was to minimize the time for the quadrotors to fly through the whole tracks while keeping the inputs within pre-defined boundaries, which results in a bang-bang profile that the control inputs spend more time at the boundaries [4, 17]. As stated before, if the input boundaries are set exactly the same as the quadrotor's physical limits, the generated trajectories will leave no margin to handle the deviation caused by disturbance, model inaccuracy and sensor imperfections, etc and artificially setting lower boundaries may lose the optimality. Thus, to solve this problem we introduce an input scaling factor ${}_ie$ for each quadrotor i to adjust the inputs' boundaries.

$$0 \leq {}_ie \leq 1 \quad (7)$$

when ${}_ie = 1$, the optimizer will take advantage of the full throttle of the quadrotors and the generated trajectories are the most aggressive but the quadrotors are vulnerable to the disturbance as they have no ability to handle the deviation. Now the input constraints of the original problem can be written as

$${}_ie u_{min} \leq {}_iu \leq {}_ie u_{max}. \quad (8)$$

The scaling factors ${}_ie$ serve as the optimization variables in our optimization problem. Thus, the optimizer can limit the inputs' boundaries by adjusting ${}_ie$ and at last ${}_ie$ for each quadrotor will converge to constant values to make the optimization objective minimal, which will be explained later.

3.3 Thrust Change Rate Constraints

According to [17], the time-optimal trajectory of a quadrotor has a bang-bang structure which is very difficult to be tracked by a quadrotor. Because the actuators have their inertia, it is impossible for rotors to achieve the required RPMs immediately. To model this property, we add the thrust's changing rate to avoid the thrust's 'step' change which is impossible to realize in the real world.

$$-\dot{u}_{max} \leq \frac{u_{k+1} - u_k}{\Delta t} \leq \dot{u}_{max} \quad \forall k \in [0, N-1], \quad (9)$$

where $\dot{u}_{max}$ is the maximum thrust changing rate and Δt is the time interval between to nodes.

3.4 Multi-objective Optimization Target

The target of the optimization problem in this article is to generate time-optimal trajectories while leaving some control margin for disturbance rejection. However, minimum time and the inputs' lowest upper boundary are contradictory. Thus, the optimization object can be written as

$$\min_{\mathbf{x}} J = J_t + J_e \quad (10)$$

where

$$J_t = \sum_i^Q S({}_i\lambda)$$

is an index measuring the total flying time of the quadrotors, which is the sum of ${}_i\lambda$ matrices of all quadrotors and

$$J_e = \sum_{i=0}^Q {}_ie$$

is an index indicating the input boundaries of all quadrotors. However, in most cases $J_t \gg J_e$. In order to optimize two terms equally, here we introduce two functions $\mathcal{F}(\cdot)$ and $\mathcal{G}(\cdot)$ to normalize these two terms. Then the target in (10) can be written as

$$\min_{\mathbf{x}} J = \alpha \mathcal{F}(J_t) + (1 - \alpha) \mathcal{G}(J_e) \quad (11)$$

where α is the weighting factor. The selection of normalized functionals $\mathcal{F}(\cdot)$ and $\mathcal{G}(\cdot)$ directly affects the optimization results and the iterations, which will be analyzed and compared in detail in Section 4.

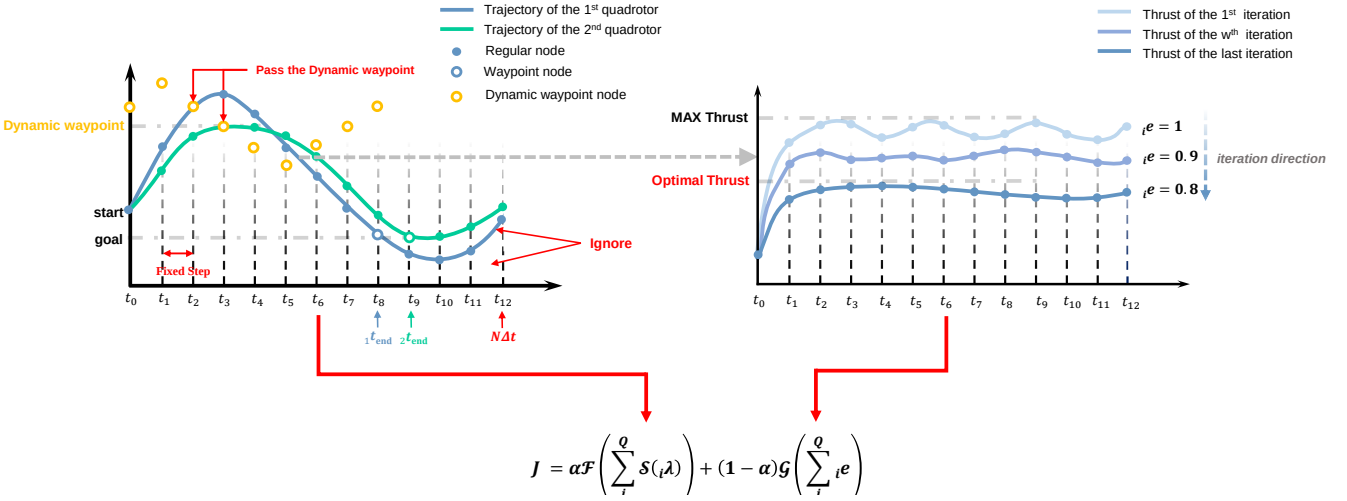


Figure 1: Sketch of the proposed method. The time step is fixed so that it is possible to add collision constraints for multiple quadrotors. The left figure shows the collision constraint and the dynamic waypoint constraint. The trajectory of dynamic waypoints is shown as yellow circles. The right figure shows that the maximum thrust decreases with the optimization. The optimization objective is to minimize the sum of the $i\lambda$ and ie while satisfying the dynamics constraints and collision constraints, etc.

3.5 Optimization Problem Summary

The optimization objective is to minimize the flying time and the input boundaries, which can be expressed as

$$\min_{\mathbf{X}} J = \alpha \mathcal{F} \left(\sum_i^Q \mathcal{S}(i\lambda) \right) + (1 - \alpha) \mathcal{G} \left(\sum_i^Q i e \right) \quad (12)$$

where Q is the number of quadrotors in the swarm. $\mathbf{X}$ is a set consisting of the optimization variables $i\mathbf{x}$ of all the Q quadrotors, and $i\mathbf{x} = [ie, i\mathbf{x}_0, i\mathbf{x}_1, \dots, i\mathbf{x}_{N-1}]$. Note that all nodes of each drone share the same energy variable ie in the state variables $i\mathbf{x}$, which is different from other state variables are redefined at each node $i\mathbf{x}_k$.

4 NUMERICAL SOLUTION AND ANALYSIS

4.1 Numerical simulation setup

We utilize the nonlinear solver CasADi[18] to solve the optimization problem described in Section 3. A 6-waypoint track is designed to test the performance of the proposed method. The position of the waypoints is listed in Table 1. It should be noted that the second waypoint is a dynamic waypoint whose position changes in y direction by

$$\mathbf{P}_1^{wj}(t) = \begin{bmatrix} x_1(t) \\ y_1(t) \\ z_1(t) \end{bmatrix} = \begin{bmatrix} 25 \\ 5 + 6 \sin(2\pi t/1.5) \\ 3 \end{bmatrix}$$

or

$$\mathbf{P}_2^{wj}(t) = \begin{bmatrix} x_2(t) \\ y_2(t) \\ z_2(t) \end{bmatrix} = \begin{bmatrix} 25 \\ 5 + 3 \sin(2\pi t/6) \\ 3 \end{bmatrix}$$

to show the performance of the proposed method with different dynamic waypoints' speeds. 5 quadrotors ($Q = 5$) are used in the optimization problem. In the following part of this section, we will discuss the normalization functions and the weighting factors in detail.

Table 1: The position of the waypoints

waypoint NO.	$x[m]$	$y[m]$	$z[m]$
1	5	15	2
2	25	$5 + y_*(t)$	3
3	20	25	5
4	14	14	2
5	18	18	6
6	5	14	4

4.2 normalization function

We select the normalization function $\mathcal{G}(\cdot)$ for J_e as

$$\mathcal{G}(J_e) = \frac{1}{Q} J_e \quad (13)$$

since

$$J_e \in [0, Q] \quad (14)$$

so that

$$\mathcal{G}(J_e) \in [0, 1] \quad (15)$$

We also have two normalization functions $\mathcal{F}(\cdot)$ as the candidates to map the J_t term in equation (12) to $[0, 1]$ which are sigmoid function

$$\mathcal{F}_{sigmoid}(J_t) = \frac{\exp \mathcal{V}(J_t)}{1 + \exp \mathcal{V}(J_t)} \quad (16)$$

where

$$\mathcal{V}(J_t) = 0.005(J_t - V_m)$$

and linear mapping function

$$\mathcal{F}_{linear}(J_t) = \frac{1}{2(V_m - V_o)} \frac{1}{Q} J_t - \frac{V_o}{2(V_m - V_o)} \quad (17)$$

where

$$V_m = N \sum_{i=0}^{N_{wp}} \frac{i}{N_{wp}}$$

represents that the quadrotor passes through N_{wp} waypoints with the same time interval and

$$V_o = \frac{v_{min}}{v_{max}} V_m$$

is the value obtained by using the ratio of the maximum velocity v_{max} and minimum velocity v_{min} as the scaling factor.

To control the effects introduced by other variables, we set $\alpha = 0.8$ to check the optimization performance with different $\mathcal{F}(\cdot)$. The result is shown in Table 2.

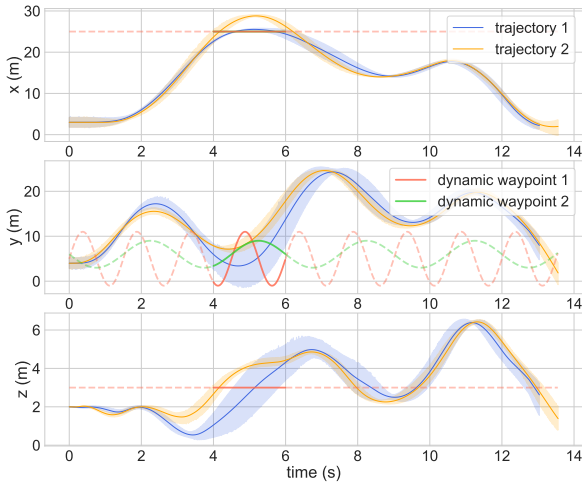


Figure 2: Trajectories of 5 quadrotors with two different dynamic waypoints. The trajectories of fast and slow dynamic waypoints are shown in red and green curves respectively.

Both functions can be guaranteed to be differentiable within the range of values. As shown in Table 2, the smoother normalization function, sigmoid, achieves better results in

Table 2: Two different normalization functions

Function type	sigmoid function	linear function
Iterations	7165	13162
Objective value	$4.02e-2$	$9.05e-1$
Single iteration time (s)	71.94	0.73
Total time (s)	515450	9608

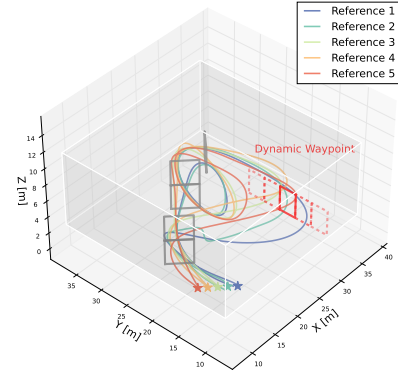


Figure 3: The generated optimal trajectories for 5 quadrotors

fewer iterations than the lower objective function value. However, its total time is hundreds of times of the linear function. Thus, we select the linear function as the normalization function. The generated optimal trajectories with $\mathcal{F}(J_t) = \frac{1}{2(V_m - V_o)} \frac{1}{Q} J_t - \frac{V_o}{2(V_m - V_o)}$ and $\alpha = 0.8$ is shown in Figure 2 and Figure 3.

4.3 Weighting factor

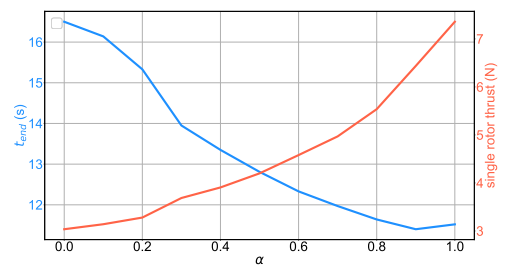


Figure 4: The flying time and maximum thrust with different α

In this section, we do a parameter study of α to show how this weighting factor influences the optimization result. We solve the optimization problem (12) with varying α and the track in Table 1 for 5 quadrotors and the result is shown in Figure 4

As shown in Figure 4, as α gradually increases, the flying time of the quadrotors decreases while the maximum thrust

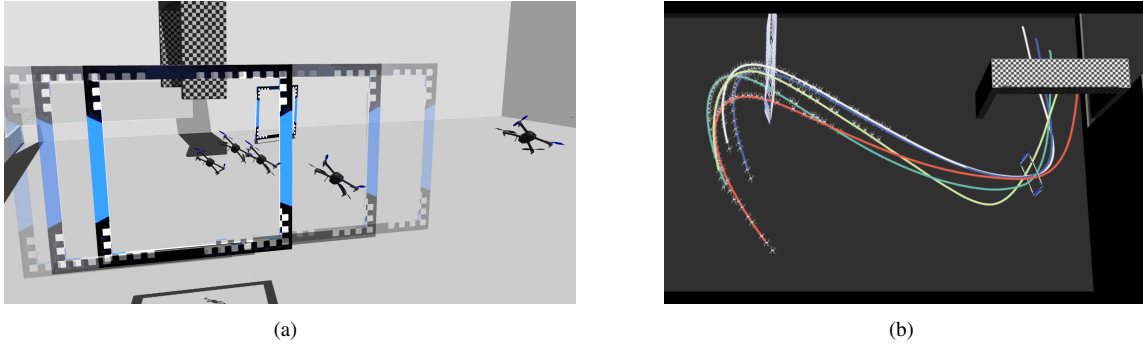


Figure 5: Gazebo simulation environment. 5 quadrotors are flying through the dynamic gate.

increases as discussed in Section 3. Figure 4 gives us a rough idea of how to tune α for some specific application scenarios.

5 SIMULATION AND REAL-WORLD FLIGHT EXPERIMENT

We first verify the proposed method in Gazebo simulation environments. The racing gates (waypoints) are deployed according to Table 1. The optimization result is used as the reference trajectories to be tracked by the quadrotors. A nonlinear model predictive controller (NMPC) is used for trajectory tracking and the control structure is shown in Figure 6. The model used in the NMPC is a 9-state quadrotor model whose states are the quadrotor's position, velocity, quaternions while the inputs are the angular velocities of the quadrotor. The angular velocity controller is a default PX4 PID controller for tracking the commands provided by the NMPC. The ACADO software is used to implement the NMPC algorithm. The readers are referred to [19] for the implementation details.

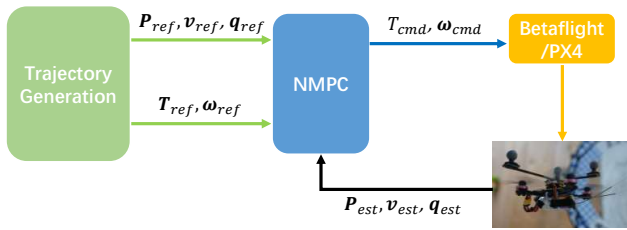


Figure 6: The control structure for the trajectory tracking. The nonlinear model predictive controller is used to track the reference trajectory. The PX4 is used as a low-level controller in simulations while the Betaflight is used in the real-world experiments.

The simulation results show that the proposed method can generate aggressive trajectories for a swarm of racing drones to fly through a pre-defined track including a dynamic waypoint. Also, as we use α to leave control margins for disturbance and model inaccuracy.

For the real-world experiments, we develop a new micro quadrotor that only weights 100g as shown in Figure 7. This



Figure 7: The 100g flying platform used in the experiment.

quadrotor runs the Betaflight onboard to provide an angular rate loop controller as it has a very accurate and stable angular velocity tracking performance. It receives angular velocity commands and throttle commands from the high-level controller. The high-level controller NMPC runs on a laptop. The onboard autopilot sends the IMU measurements and the attitude estimation to the laptop using the MAVLink protocol via the onboard WiFi module and the Optitrack motion capture system also sends the quadrotors' position measurements to the laptop. The NMPC optimizes the control inputs, the angular rates and the throttle in our case, and sends them to the onboard autopilot via the WiFi module to control the quadrotors to track the trajectories.

Due to the very limited size of the flying arena $4m \times 2m \times 2m$, we use two quadrotors to give a flying example to demonstrate the performance of the proposed method in the real world. The virtual dynamic waypoint moves in y direction by $\mathcal{P}_1^{wj}(t)$. The virtual dynamic waypoint and the trajectories of the quadrotors are shown in Figure 8.

From Figure 8, it can be seen that in the real world, the proposed method can generate the optimal trajectories for a swarm of quadrotors to fly through dynamic waypoints and also minimize the input boundaries to leave margin for the disturbance and model inaccuracy.

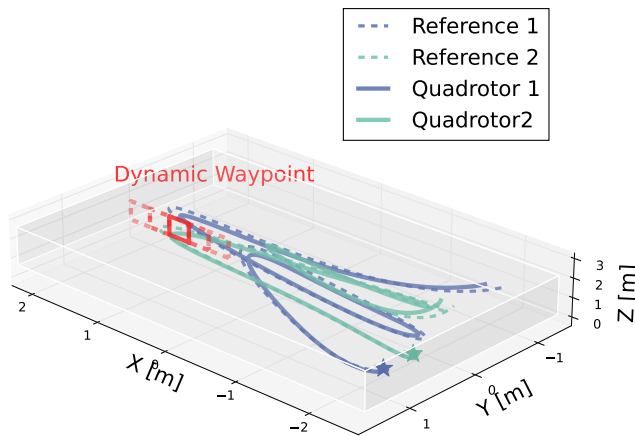


Figure 8: The collision-free trajectories of two quadrotors in the real-world experiment. The trajectory of the dynamic waypoint as shown in dashed red gates

6 CONCLUSION

In this article, we proposed a novel trajectory generation method for a swarm of quadrotors that considers the inputs' boundaries in the optimization objective. So that the generated optimal trajectories leave control margins for rejecting the deviation caused by the disturbance, model inaccuracy and sensors' imperfection. Furthermore, we also take dynamic waypoints into consideration in our optimization problem so that the quadrotors can fly through them with the optimal trajectories. Furthermore, the simulation and real-world flight results valid that the proposed trajectory generation method can guide a swarm of quadrotors flying through the waypoints (including dynamic waypoints) with the optimal index which is not only time optimal but also leaves control margins for the deviation correction.

REFERENCES

- [1] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525, 2011.
- [2] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, 31(5):664–674, 2012.
- [3] Adam Bry, Charles Richter, Abraham Bachrach, and Nicholas Roy. Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments. *The International Journal of Robotics Research*, 34(7):969–1002, 2015.
- [4] Philipp Foehn, Angel Romero, and Davide Scaramuzza. Time-optimal planning for quadrotor waypoint flight. *Science Robotics*, 6(56):eabh1221, 2021.
- [5] Gao Tang, Weidong Sun, and Kris Hauser. Learning trajectories for real-time optimal control of quadrotors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3620–3625. IEEE, 2018.
- [6] Shuo Li, Ekin Öztürk, Christophe De Wagter, Guido CHE De Croon, and Dario Izzo. Aggressive on-line control of a quadrotor via deep network representations of optimality principles. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6282–6287. IEEE, 2020.
- [7] Sebastien Origer, Christophe De Wagter, Robin Ferede, Guido CHE de Croon, and Dario Izzo. Guidance & control networks for time-optimal quadcopter flight. *arXiv preprint arXiv:2305.02705*, 2023.
- [8] Alex Kushleyev, Daniel Mellinger, Caitlin Powers, and Vijay Kumar. Towards a swarm of agile micro quadrotors. *Autonomous Robots*, 35(4):287–300, 2013.
- [9] Wolfgang Hönig, James A Preiss, TK Satish Kumar, Gaurav S Sukhatme, and Nora Ayanian. Trajectory planning for quadrotor swarms. *IEEE Transactions on Robotics*, 34(4):856–869, 2018.
- [10] KN McGuire, Christophe De Wagter, Karl Tuyls, HJ Kappen, and Guido CHE de Croon. Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. *Science Robotics*, 4(35):eaaw9710, 2019.
- [11] Carlos E Luis, Marijan Vukosavljev, and Angela P Schoellig. Online trajectory generation with distributed model predictive control for multi-robot motion planning. *IEEE Robotics and Automation Letters*, 5(2):604–611, 2020.
- [12] Bardienus P Duisterhof, Shushuai Li, Javier Burgués, Vijay Janapa Reddi, and Guido CHE de Croon. Sniffy bug: A fully autonomous swarm of gas-seeking nano quadcopters in cluttered environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9099–9106. IEEE, 2021.
- [13] Xin Zhou, Xiangyong Wen, Zhepei Wang, Yuman Gao, Haojia Li, Qianhao Wang, Tiankai Yang, Haojian Lu, Yanjun Cao, Chao Xu, et al. Swarm of micro flying robots in the wild. *Science Robotics*, 7(66):eabm5954, 2022.
- [14] Yuyang Shen, Jinming Xu, Jin Zhou, Danzhe Xu, Fangguo Zhao, Jiming Chen, and Shuo Li. Aggressive trajectory generation for a swarm of autonomous racing drones. *arXiv preprint arXiv:2303.00851*, 2023.

- [15] Hyungpil Moon, Jose Martinez-Carranza, Titus Cieslewski, Matthias Faessler, Davide Falanga, Alessandro Simovic, Davide Scaramuzza, Shuo Li, Michael Ozo, Christophe De Wagter, et al. Challenges and implemented technologies used in autonomous drone racing. *Intelligent Service Robotics*, 12:137–148, 2019.
- [16] Ziyu Zhou, Gang Wang, Jian Sun, Jikai Wang, and Jie Chen. Efficient and robust time-optimal trajectory planning and control for agile quadrotor flight. *arXiv preprint arXiv:2305.02772*, 2023.
- [17] Dharmesh Tailor and Dario Izzo. Learning the optimal state-feedback via supervised imitation learning. *Astrodynamics*, 3:361–374, 2019.
- [18] Joel AE Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11:1–36, 2019.
- [19] Davide Falanga, Philipp Foehn, Peng Lu, and Davide Scaramuzza. Pampe: Perception-aware model predictive control for quadrotors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

MTSO: Multi-Target Search Optimisation based on Probability Map

Aman Virmani

Department of Electrical Engineering
Delhi Technological University
Delhi, India
virmaniaman4@yahoo.com

Vayam Jain

Department of Applied Physics
Delhi Technological University
Delhi, India
jainvayam@gmail.com

Nilesh Aggarwal

Department of Applied Physics
Delhi Technological University
Delhi, India
nilesh.ggoo@gmail.com

Arjun Gupta

Department of Electronics Engineering
Delhi Technological University
Delhi, India
arjun222gupta@gmail.com

Anunay

Department of Mechanical Engineering
Delhi Technological University
Delhi, India
anunay2608@gmail.com

Dr. Anup Kumar Mandpura

Department of Electrical Engineering
Delhi Technological University
Delhi, India
kanup@dtu.ac.in

Abstract—This report presents an optimized, decentralized way of searching for targets using swarms of rotary based unmanned aerial vehicles with information available from onboard EO/IR sensors. The are three main objectives of the proposed algorithm: time-optimized multi-target search, maximum and optimized payload drop, and maximum area coverage. The real-time application of these UAVs is in Human Aid and Disaster Relief scenarios mostly where the survivors have to be identified and given relief material. The proposed controller is inspired by the multi-target particle swarm optimized method(MTPSO) and its limitations of convergence towards Pbest and Gbest which may not be the best option in HADR scenarios. The algorithm involves the target probability distribution of the cells present in the target area which keeps on updating dynamically as the UAVs dive deeper into the target area. The deep learning-based model for target detection is deployed on each UAV using a dual camera with a limited field of view and its target discriminability varies as a function of the Class of the target, environmental conditions, etc. These parameters are given input to the probability distribution method for generating a probability map using which the UAVs optimize their path. The algorithm is evaluated on Ardupilot's SITL platform for parameter tuning and simulation in various scenarios which are later compared with existing search methods.

Keywords—Swarm, Unmanned Aerial Vehicles, Multi-target Search, Multi-agent systems, Particle Swarm Optimization, Probability map, k-means, Hungarian algorithm.

I. INTRODUCTION

The demand for Unmanned Aerial vehicles has increased in the past years, as they are capable of rapidly covering areas, and providing better surveillance and efficient monitoring areas which gives them an edge over other alternatives. Because of these characteristics, the role of UAVs in civilian and military applications has become popular. The ground target search problem is one of the most important and popular applications of UAVs. Over time, the problem has become more complex as the number of targets and the search area has grown; to solve these huge UAV swarms are used. In this type of mission, it is essential to provide aid quickly and find multiple targets as quickly as possible.

There have been multiple solutions[1] put forward by various researchers for the problem of multi-target acquisition. The most common is a simple exhaustive search, i.e. scanning the complete area using pre-defined trajectories. Although used successfully, this method has certain limitations. First, as the trajectories are pre-fed, they fail to adapt according to the dynamic situations, resulting in more time taken. Second, as the decisions are not being made autonomously, there is a need for the operations team to look after the mission which is both expensive and time-consuming.

There are also unique approaches based on heuristic methods like multiple target search area optimization [2] which have proven to give good results during operations. The algorithm begins with an exhaustive search, identifies the global best and personal best during the search, and changes their behavior according to them. This algorithm satisfies all three main objectives but has its limitations like the guidance of UAVs away from multiple targets if the global best is identified somewhere else. Also, the percentage of the area covered can be improved if additional lines of UAVs are added.

There are also other centralized approaches similar to our algorithm which involves dividing the target area into small n-cells [3],[4] and assigning a probability to each cell. As the UAVs move further into the search area the probability of the cells keeps on updating depending on the targets found by them. This creates a dynamic updating probability map of the search area. This method adapts according to the situation but has its limitations. First, the centralized controller stores data from all agents and this requires a robust communication architecture.

In this report, we propose a decentralized swarm controller which is based on the probability method which has been proven to give good results but its application in multi-target searches has been limited due to a lack of adequate structure, multi-UAV collisions, and poor navigation in unfamiliar search locations. The proposed controller has a defined structure to optimize search and payload drops and incorporate inter-UAV collision avoidance. The algorithm is scalable, and fault-tolerant but is computationally expensive and hence requires a strong OnBoard Computer. In our case, we chose the NVIDIA Jetson Nano as the OBC as the requirement was to deploy a CPU-intensive algorithm for search and GPU intensive deep learning-based model for object detection. Also, for the communication architecture requirement, we integrated a software-defined Radio with mesh topology to test the controller in a real-world environment.

The limitation of the probability-based method was the need for a centralized controller so the probability map is the same for all the UAVs, but the current controller estimates velocity based on the weight assigned to the cells making the process dynamic and real-time hence eliminating the issue of uniformity. There are two main terms in the velocity vector term, first the inertial term which controls the factor exploration vs exploitation, and the second the derivative of the probability distribution curve. The weights of these were tuned visually such that there was no saturation in the local minima of the probability distribution curve. The exploration and exploitation characteristics are also analyzed over different functions which have given different performances and can be used for different missions. [5]-[11].

II. PRELIMINARIES

2.1 Problem Formulation

We envision a swarm of quadcopter UAVs moving across a search region specified by a starting waypoint W_{start} , an ending waypoint W_{end} , and a breadth B , seeking numerous targets strewn throughout the landscape. Because all of the UAVs are at the same altitude, they all travel in the same direction. The position vector $p_i^t = [x_i, y_i]^T$ and the velocity vector $v_i^t = [\dot{x}_i, \dot{y}_i]^T$ where N is the number of UAVs, may be employed for the UAV i ($i = 1, 2, \dots, N$) at time t .

Using an onboard software stack. Each UAV has an onboard camera that can identify and locate objects inside its field of vision (FOV). Each UAV is also equipped with a single payload that may be dropped at one of many different target locations. The ultimate purpose of UAVs is to scan the surveillance area, find as many targets as possible, drop as many payloads as possible, and reach the end of the search area while avoiding inter-UAV collisions.

A partially connected ad-hoc network [12] is used to mimic the network architecture of all the agents. The model can be represented mathematically by an undirected graph $G = (V, E)$, where $V = 1, 2, \dots, N$ is the set of nodes and $E = \{(i, j): i, j \in V, \|p_i - p_j\| \leq DC\}$ is the edge set. A single UAV directly connects only to its neighboring agents denoted by $NG = \{j \in V, (i, j) \in E\}$

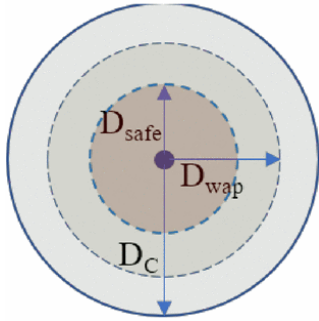


Fig.1: Pictorial representation of D_{safe} , D_{wap} , and D_C

Agents only share the bare minimum of data, such as their location data for inter UAV-collision avoidance, the GPS location and probability of detected targets for cooperative convergence on target clusters, and their payload status for smart and cooperative payload delivery a pictorial representation of the radii are given in fig.1..

The following bounding requirements (in order of priority) confine the algorithm:

- Avoid colliding with other UAVs.
- Identify every target in the search area
- Drop as many gooes as feasible as fast as possible.

2.2 Probability Map Formulation

Initially the search area is divided into N -cells, where the dimension of each cell is chosen on the basis of the computational capability of the Onboard Computer of the UAV. In our case, the NVIDIA Jetson nano can support up to 30,000-40,000 grid cells, hence according to this limitation the cell dimension is decided.

An object detection algorithm has been used to execute aerial detection of the objects using the images from the EO/IR camera. After deploying and testing the leading models for object detection, YOLOV5 was chosen for its high accuracy while also maintaining decent FPS when deployed on the Jetson Nano. You Only Look Once (YOLO), takes in the images from the EO/IR

sensor and predicts the classes and the bounding boxes along with each detection's confidence value. The object is subsequently geotagged using the field of view of the EO/IR sensor.

The probability and area of influence are the two inputs that are estimated for the construction of a probability map. The probability of a target being at the specific cell can be specified by the confidence level that the YOLOv5 model returns and has a range of (0.7,1). The area of influence on the other hand has various factors which need to be accounted for and have been discussed below briefly.

- **Class of the Object:-** There are multiple types of targets that we took in our test case e.g. humans, fuel dumps, type-A vehicles, and type-B vehicles. Certain classes such as Type A vehicles are physically larger than other objects such as Humans and easily detected. Moreover, they should have a much higher area of influence than humans as the presence of vehicles implies a greater concentration of possible targets.
- **Altitude and Size of the object:-** For a given class the relationship between the altitude of the UAV and the size of the object in the image plane (in pixels) is directly proportional and can be calculated using the field of view of the camera system and the resolution of the camera using basic trigonometry.
- **Cluster of targets:-** In real-world environments targets tend to stick together and travel in clusters, hence the probability of a specific grid cell needs to be determined increased if the other targets are also dedicated in its vicinity. This can be represented mathematically just by adding the probability over the cells again.

III. ALGORITHM DESIGN

3.1 Initialization of UAVs

We begin the target-search method in a thorough manner, with UAVs placed in a line at the start of the search region, to optimize the initial exploration. The overlap between each UAV's FOV is limited to a bare minimum and remains consistent (O_{min}). For repeated runs, if the search region is large enough, it can be divided into many portions.

3.2 Inertial Term

The inertial velocity term, as its name implies, aids in maintaining the swarm unit's original course and prevents abrupt changes in each UAV's trajectory. It's the result of multiplying the current velocity by the inertial function. The algorithm's exploration vs. exploitation features is controlled by the inertial function, which is generally calibrated to give better exploration at first and exploitation as the swarm reaches deeper into the area and has already collected sufficient information about the area.

3.3 Probability Calculation

The concept is based on the fact that targets stick together and travel in clusters, hence a probability distribution is proposed. The mathematical model used to estimate the probability distribution is the Gaussian / Normal distribution curve which is represented below:-

$$f(x) = e^{\frac{-1}{2}(\frac{x-\mu}{\sigma})^2} \div \sigma\sqrt{2\pi}$$

where,
 σ = Standard Deviation

μ = Mean

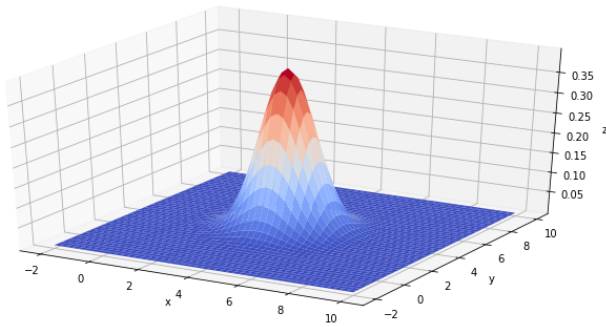


Fig.2: Graphical representation of gaussian curve

σ represents the area of influence which as specified in the previous section has to be selected according to the class of targets identified during the search. The below table represents the area of influence for each class:-

S.No	Type of target	Range of Influence
1	Human	20-40 meters
2	Fuel Dumps	40-60 meters
3	Type-B vehicles	60-100 meters
4	Type-A vehicle	100-140 meters

Table 1: Limits of σ for the type of class

The reason for taking the limits was the second factor which corresponds to increase in probability if the number of pixels occupied by that class of target is more. For YOLOV5 there is a relationship between the number of pixels occupied by the target and the accuracy of the detection given as:

$$m(x) = \frac{1}{1 + e^{s(x-m)}}$$

Where:-

$f(x)$ = Accuracy of the detection

x = Size of the object

s = Steepness

m = Shift

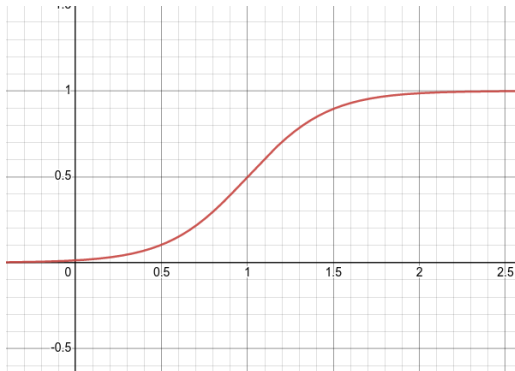


Fig.3: The S-shaped curve of $m(x)$

The graph for the above function $m(x)$ is shown in fig.3.

The YOLOV5 model was tested and the value of the hyperparameters came out be:

$S = -0.1$

$M = 16$

These parameters represent that the YOLOV5 model can only detect targets with good accuracy if they occupy more than 16 pixels in the input image. Now, this behavior of the graph is imposed between the limits to increase the σ if the number of pixels of the same target class increases, hence making the algorithm more operationally robust.

$$\sigma = m(x) * (Upper\ Limit - Lower\ Limit) + Lower\ Limit$$

This process is repeated for each detected object to generate an independent probability map for each target and the corresponding probability maps are added up to find the final probability map $P(x)$ countering the third point that probability needs to be increased further if more targets keep on detecting in high probable areas. This map can be visualized as a surface plot where the x and y-axis represent the search grid and the z-axis represents the probability value at that specific grid point. The map gives a representation like shown in fig.4.

$$P(x) = \sum_{i=0}^n f_i(x)$$

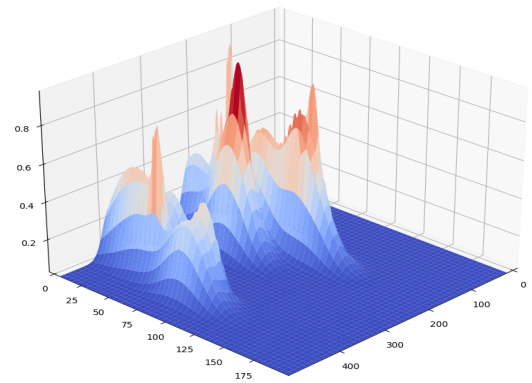


Fig.4: Gaussian-based probability distribution curve of the target area

3.4 Probability-Based Velocity

The probability-based velocity term aims to guide the UAVs towards the high probability areas using the probability map generated earlier. The partial derivative/gradient of the probability map is taken to get the following surface plots representing the partial derivative in the x and y-axis respectively as shown in fig.5.

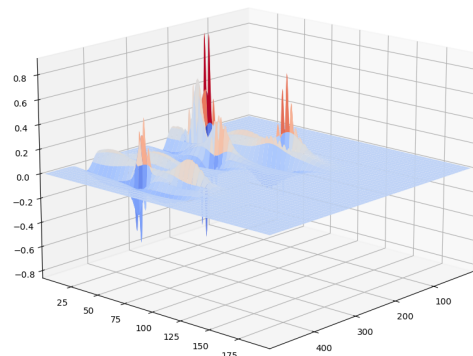


Fig.5: Partial derivative of $P(x)$ in the x and y-axis respectively

$$D(x) = \frac{dP(\sum_{i=0}^n f_i(x))}{dx}, D(y) = \frac{dP(\sum_{i=0}^n f_i(y))}{dy}$$

The probability-based velocity term uses this gradient to guide it towards the high probability areas using the formula:-

$$V_{(x,y)} = w * D(x, y)$$

where,

w = Tunable constant

3.5 Payload Optimization

The most crucial duty in the quick reaction approach is the payload drop. For job assignment and optimal delivery, we begin by assigning the UAVs to one of three states, which aids in determining which UAVs should continue to search and which should proceed for payload drop:

State	Meaning
State '1'	Payload available
State '0'	Going to deliver the payload
State '-1'	Payload has been dropped

Table 2: Payload drop states

As long as the target is within the maximum displacement range, the locations of the identified targets are continuously relayed between the UAVs, and the payload delivery job is assigned to the UAV nearest to the target.

$$v_{drop}^{t+1} = C_3 * (p_{drop} - p^t)$$

Furthermore, if the UAV detects a target B that is closer to target A while dropping a payload on target A, it moves its drop location to target B, re-qualifying target B for payload drop. While delivering the payload, the UAV continues to seek targets and communicate with other UAVs. It does not return its steps after delivering the goods, instead of continuing the hunt from the drop spot.

3.6 Inter-UAV Collision Avoidance

To maintain the safety of any swarm system, inter-UAV collision avoidance is essential. As a result, we use the consensus equation to generate an extra velocity term that may be vectorially added to each agent's final velocity [13]. If the distance between any nodes gets lower than D_{safe} , avoidance is initiated. This method is simple to include in our framework and is represented as follows:

$$D = \|p_i - p^t\|_2$$

$$v_c^{t+1} = \begin{cases} C_4 * \sum_{i=1}^N (p_i - p^t) * \left(1 - \frac{D_{safe}}{D}\right), & D \leq D_{safe} \\ 0, & D > D_{safe} \end{cases}$$

This method is suited for our application since it just considers the agents coming for collision. The D_{safe} parameter is usually set to match the maximum swarm velocity, although it can be manually adjusted if necessary.

3.7 Net Equivalent Velocity

The UAVs communicate with each other to share information about the targets and allow for the decentralized construction of probability maps, such that the final velocity vector can be calculated onboard each UAV independently. Hence the final velocity vector is as follows:

$$V = v_{Inertial} + v_{Probability} + v_{Obstacle Avoidance} + v_{Payload Drop}$$

3.8 Return Mission

As the mission in MTSO progressed a lot of empty unsearched gaps in the search area were left behind to optimize the time of the mission. Thus when the MTSO mission is over another grid search mission is plotted on the unsearched regions so as to be sure that no other targets are left behind.

3.8.1 K-Means Clustering Algorithm

In the return mission the first task is to calculate the total unsearched area left behind during the search as shown in fig.6. Once this is calculated all the points in the area are segregated in clusters using the K-means clustering algorithm. The K-Means Clustering technique divides the input dataset into multiple clusters based on their distances from the centroids. It's a clustering-based approach in which each cluster has its own centroid, which is used to characterize the clusters. The input given to the Kmeans is the coordinates of the unsearched area which are represented as d-dimensional real vectors as $(x_1, x_2, \dots, x_n)$. K means to organize the n groups to k groups where $k \leq n$. The main equation of k means is

$$\arg \min_S \sum_{i=1}^k \sum_{x \in S_i} \|x - \mu_i\|^2 = \arg \min_S \sum_{i=1}^k |S_i| \text{Var } S_i$$

where μ_i is the mean of points in S_i . This is equivalent to minimizing the pairwise squared deviations of points in the same cluster

$$\arg \min_S \sum_{i=1}^k \frac{1}{|S_i|} \sum_{x, y \in S_i} \|x - y\|^2$$

Which can be reduced to

$$S_i \left| \sum_{x \in S_i} \|x - \mu_i\|^2 = \sum_{x \neq y \in S_i} \|x - y\|^2 \right|$$

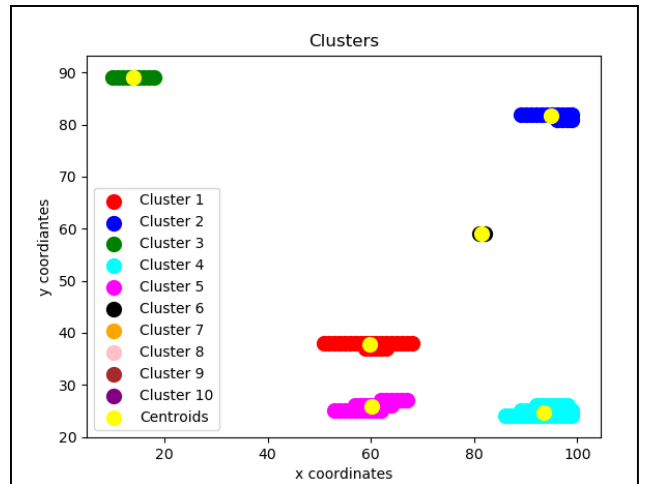


Fig.6: Image showing the division of clusters and centroids using the K-means algorithm

Because the total variance is constant, this is identical to iteratively maximizing the sum of squared deviations across points in distinct clusters.

The number of clusters in which the unsearched area is to be divided is decided by the elbow function. The elbow method runs k-means clustering on the dataset for a range of values for k and then for each value of k computes an average score for all clusters. Using these scores a graph is plotted as shown in fig.7 and the line with the highest changes in slope indicates the optimal number of clusters that need to be made.

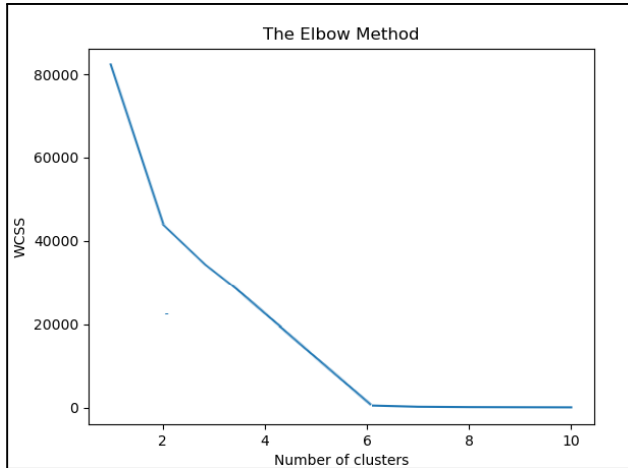


Fig.7: Elbow method showing the max slope change at 6 clusters

3.8.2 Health Function

Once the clusters are divided, UAVs are also segregated based on the priority order given below :-

- Payload dropped or not:- As the number of targets and the number of UAVs are not always equal, so in a case if the number of targets is less than the number of UAVs the UAVs that haven't dropped their payload yet are listed first for the return mission.
- Health function :- The rest of the UAVs with the highest battery percentage and proximity to targets are chosen first for the return mission.

3.8.3 Hungarian Algorithm

The clusters have been constructed, and the UAVs that are qualified for the mission have been separated; now each cluster must be assigned to a UAV that will conduct the search. The nearest possible cluster should be assigned to each UAV for mission robustness and time optimization, and the Hungarian method was utilized for this. The Hungarian Method is a technique based on the idea that if the same value is added or subtracted from every member of a matrix's row or column, the new assignment issue's optimal solution should be the same as the original problem.

3.8.4 Polygon Grid Search

The Hungarian algorithm provided each UAV its cluster. Now each UAV is to be assigned waypoints to search in its area. To assign the waypoints a self-implemented polygon grid search was invented that can work on any polygon. Our algorithm takes in the coordinate points of the boundary of the polygon and forms a perfect rectangle around the polygon completely enclosing the polygon inside it. Now the Grid search method is applied on the rectangle providing initial and final waypoints to the UAV as it

continues the mission. To overcome the extra space provided by the rectangle that is not in the polygon a line intersect function is implemented to keep the UAV inside the polygon.

IV. SIMULATION AND TESTING

This section firstly describes the methods of simulations used to test our code. Then, a comparative study of the tuning of different inertial functions according to different FOVs is given. Lastly, the effectiveness of MTSO is compared with other methods in various simulated scenarios.

A. Simulation Environment

To simulate the algorithm in various scenarios and later on compare it to other algorithms, Ardupilot SITL was utilized. To visualize the targets, a self-created mavproxy module is used, which selects the targets at random and visualizes them in the SITL map as shown in fig. 8-11. If the target is detected, the probability map is updated and presented continuously using OpenCV and matplotlib libraries. Because the SITL only accepts GPS data, conversion to UTM coordinates were conducted.

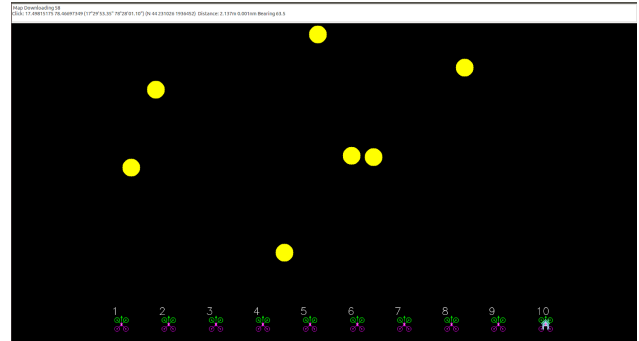


Fig.8: Simulation Environment and Initialization of UAVs

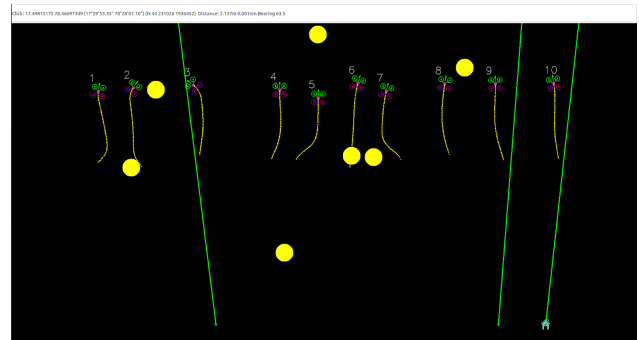


Fig.9: Convergence of UAVs on targets in search area

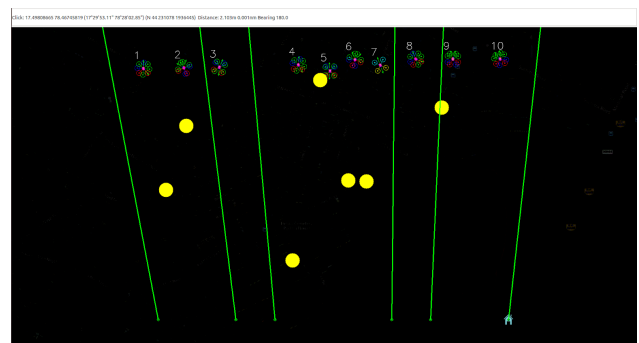


Fig.10: UAVs halting and performing clustering

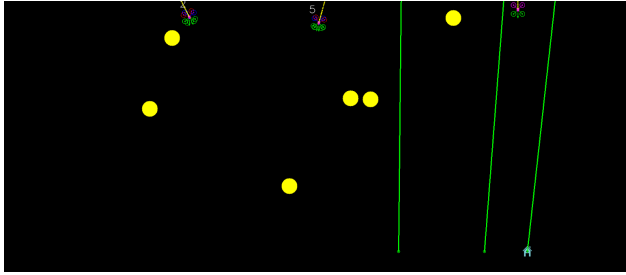


Fig.11: UAVs searching their assigned cluster according to the Hungarian algorithm

B. Path Map of the Swarm

In order to evaluate the mission later, a real-time map of each UAVs travel trajectory is created at the start of the flight. This map also shows the target that was discovered. The map is created using a straightforward openCV function and is shown with a real-time mission updates shown in fig.12.

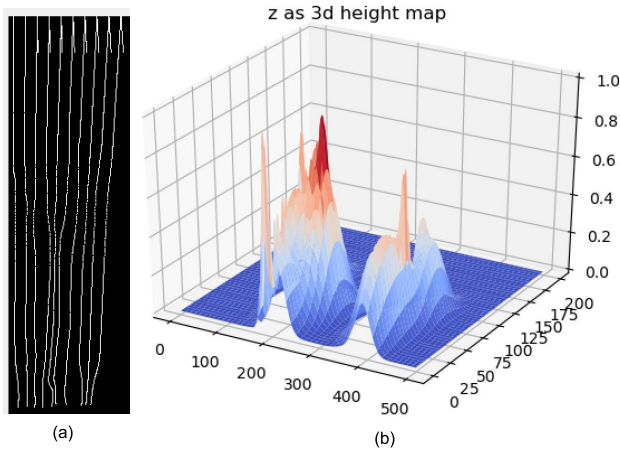


Fig.12: (a)Path map of the swarm and (b) Probability Map for the test case of FOV 140x80

C. Quantitative Comparison with Different FOVs

The algorithm of MTSO with probability map was rigorously tested against different parameters for FOV of the camera to find its performance and discrepancies if any. These tests were carried out while the camera module's FOV was changed. The FOV settings used in the experiment were chosen with the least amount of overlap in mind (0 min).

Table III summarizes the outcomes of the tests. The data clearly illustrates that expanding the FOV enhances exploration since the majority of the search region is searched, resulting in an improved overall number of discovered targets but also increasing the total time needed to scan the environment. So it is essential to find an optimum balance of efficiency, time optimization and total area scanned to get the best results for the mission.

Metrics	FOV (in meters)		
	55X32	70 X40	140X80
Target Detected	15	17	20
Payload Dropped	14	17	19

Time For Mission(s)	188.14	182.24	176.89
Total Area Scanned	92.46	96.23	98.91

Table 3: Comparison between different FOVs

D. Quantitative Comparison with other approaches

This part shows the comparison between MTPSO, Exhaustive Search and our approach. Below in table.4 is a summary of the results of different qualities of the search algorithm by changing fov in different scenarios by all three algorithms.

Metrics	Method	FOV		
		55X32	70X40	140X80
Targets Detected	Exhaustive Search	19	20	20
	MTPSO	15	16	19
	New Approach	15	17	20
Payload Dropped	Exhaustive Search	18	18	19
	MTPSO	15	15	17
	New Approach	14	17	19
Time For Mission	Exhaustive Search	402.63	422.75	454.56
	MTPSO	207.86	200.01	193.45
	New Approach	188.14	182.24	176.89
Total Area Scanned	Exhaustive Search	99.99	99.99	99.99
	MTPSO	92.61	96.38	98.32
	New Approach	92.46	96.23	98.91

Table 4: Performance metrics evaluated for different approaches with different FOVs

In each of the scenarios outlined above, MTSO with probability map outperformed exhaustive search and MTPSO. The time used by MTSO was roughly half that of the exhaustive search in terms of time optimization. Because the probability map aids in swarm convergence rather than selecting a Pbest and Gbest, the convergence towards the goal is finer, resulting in a smaller scanned region. The processing power required by MTSO utilizing a probability map is significantly lower than that required by any other probability graph approach, and it is easily implemented in a decentralized version on very light hardware platforms with minimum connectivity.

Hyper parameter	Value	Hyper parameter	Value
D_{wap}	20m	v_{min}	0m/s
D_c	500m	N	20
D_{max}	2.5*FOV	D_{safe}	15m
N_s	5	w	30
α	105°	μ	0
v_{max}	15m/s	m	16

Table 5: Hyperparameters used for simulations**V. CONCLUSION**

The new algorithm, unlike the earlier MTPSO algorithm, does not limit the movement of UAVs on the basis of only PBEST and GBEST, which often led to UAVs getting stuck in local minima and being unable to contribute for the rest of the mission. The new approach allows the movement of the UAVs on the basis of all the detected objects using a probability map, hence areas with a much higher concentration of objects take precedence. Moreover, the impact of each object is not constant unlike the previous approaches and various factors like Altitude, Class and Model Confidence which actually affect the ideal behavior in real-world environments are taken into account for assigning the probability which leads to better convergence of the swarm. This also allows for considerably more payloads to be dropped quickly, when compared to exhaustive search and MTPSO increasing the effectiveness of the UAV swarms in real-world disaster relief scenarios. To overcome the limitation of the new algorithm of limited coverage of the search area a return mission is proposed which can be used when the endurance of the UAVs allows, to exhaustively search the left out areas and confirm any objects missed by the initial pass of the algorithm.

The various hyperparameters used in the new algorithm have been tuned manually but instead, they can be tuned using advanced techniques like evolutionary hyperparameter tuning which can further improve the performance of the algorithm.

In future work, the research aims to improve:-

- Refining the revert mission to do an iterative MTSO.
- Further improving MTSO to decrease time and increase efficiency.

REFERENCES

- [1] Cabreira, T.M.; Brisolara, L.B.; Ferreira Jr., P.R. Survey on Coverage Path Planning with Unmanned Aerial Vehicles. *Drones* 2019, 3, 4.
- [2] L. Bertuccelli and J. How, "Robust UAV search for environments with imprecise probability maps," in *Proc. IEEE Conf. Decision Control*, Dec. 2005, pp. 5680–5685.
- [3] Y. Yang, A. Minai, and M. Polycarpou, "Decentralized cooperative search by networked UAVs in an uncertain environment," in *Proc. Amer. Control Conf.*, Jun.–Jul. 2004.
- [4] J. Hu, L. Xie, K. Lum and J. Xu, "Multiagent Information Fusion and Cooperative Control in Target Search," in *IEEE Transactions on Control Systems Technology*, vol. 21, no. 4, pp. 1223–1235, July 2013.
- [5] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, pages 1942–1948, IEEE Press, Piscataway, NJ, 1995.
- [6] Y. Shi and R. Eberhart., "A modified particle swarm optimizer", In *Evolutionary Computation Proceedings*, 1998. *IEEE World Congress on Computational Intelligence.*, The 1998 IEEE International Conference on, pages 69–73.
- [7] IEEE, 2002. R.C. Eberhart and Y. Shi., "Tracking and optimizing dynamic systems with particle swarms", In *Evolutionary Computation*, 2001. *Proceedings of the 2001 Congress on*, volume 1, pages 94–100.
- [8] IEEE, 2002. A.Nikabadi, M.Ebadzadeh , "Particle swarm optimization algorithms with adaptive Inertia Weight : A survey of the state of the art and a Novel method", *IEEE Journal of evolutionary computation* , 2008.
- [9] R.F. Malik, T.A. Rahman, S.Z.M. Hashim, and R. Ngah, "New Particle Swarm Optimizer with Sigmoid Increasing Inertia Weight", *International Journal of Computer Science and Security (IJCSS)*, 1(2):35, 2007.
- [10] J. Xin, G. Chen, and Y. Hai., "A Particle Swarm Optimizer with Multistage Linearly-Decreasing Inertia Weight", In *Computational Sciences and Optimization*, 2009. *CSO 2009. International Joint Conference on*, volume 1, pages 505–508. IEEE, 2009.
- [11] Y. Feng, G.F. Teng, A.X. Wang, and Y.M. Yao., "Chaotic Inertia Weight in Particle Swarm Optimization", In *Innovative Computing, Information and Control*, 2007. *ICICIC'07*.
- [12] I. Bekmezci, O. K. Sahingoz, and Ş. Temel, "Flying AdHoc Networks (FANETs): A survey," *Ad Hoc Networks*, vol. 11.
- [13] Anuj Agrawal, Aniket Gupta, Joyraj Bhowmick, Anurag Singh, Raghava Nallanthigal, "A Novel Controller of Multi-Agent System Navigation and Obstacle Avoidance", *Procedia Computer Science*, Volume 171, 2020, Pages 1221–1230.
- [14] A. Gupta, A. Virmani, P. Mahajan and R. Nallanthigal, "A Particle Swarm Optimization-Based Cooperation Method for Multiple-Target Search by Swarm UAVs in Unknown Environments," 2021 7th International Conference on Automation, Robotics and Applications (ICARA), 2021, pp. 95–100, doi: 10.1109/ICARA51699.2021.9376529.
- [15] N. Aggarwal, Anunay, V. Jain, T. Singh and D. K. Vishwakarma, "DLVS: Time Series Architecture for Image-Based Visual Servoing," 2023 8th International Conference on Control and Robotics Engineering (ICCRE), Niigata, Japan, 2023, pp. 101–107, doi: 10.1109/ICCRE57112.2023.10155585.
- [16] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks*, 4, 1942–1948. DOI: 10.1109/ICNN.1995.488968.
- [17] Coello Coello, C. A., Lamont, G. B., & Van Veldhuizen, D. A. (2007). *Evolutionary algorithms for solving multi-objective problems* (2nd ed.). Springer.
- [18] Tan, Y., Zhu, J., & Ding, X. (2013). A hybrid multi-objective particle swarm optimization algorithm. *Soft Computing*, 17(6), 1013–1031. DOI: 10.1007/s00500-013-0983-3.
- [19] Zeng, J., Chen, J., Zhang, Q., & Zhang, B. (2018). Multi-objective particle swarm optimization with decomposition for optimization problems with variable linkages. *IEEE Transactions on Evolutionary Computation*, 23(2), 217–231. DOI: 10.1109/TEVC.2018.2817386.
- [20] López-Ibáñez, M., Paquete, L., & Stützle, T. (2010). Hybridization of evolutionary multiobjective algorithms: A survey. *ACM Computing Surveys (CSUR)*, 42(4), 1–37. DOI: 10.1145/1830761.1830763.
- [21] N.Roberto, Z.Qian, L.Yanmin H.Huayao, Y.Meilan, S.Xiaoli, 02/03/2022 "Multi-Objective Particle Swarm Optimization with Multi-Archiving Strategy", 1058–9244, doi:10.1155/2022/7372450.

- [22] Raida SELLAMI, Farooq SHER, Rafik NEJI, An improved MOPSO algorithm for optimal sizing & placement of distributed generation: A case study of the Tunisian offshore distribution network (ASHTART), Energy Reports, Volume 8, 2022, Pages 6960-6975, ISSN 2352-4847, <https://doi.org/10.1016/j.egy.2022.05.049>.
- [23] D.Van, W.Shihua, L.Yanmin, Z.Kangge, L.Nana, W.Yaowei, 2022, "Multiobjective Particle Swarm Optimization Based on Ideal Distance", 2022, DOI:10.1155/2022/3515566
- [24] Yang M, Liu Y, Yang J. A Hybrid Multi-Objective Particle Swarm Optimization with Central Control Strategy. Comput Intell Neurosci. 2022 Mar 9;2022:1522096. doi: 10.1155/2022/1522096. PMID: 35310587; PMCID: PMC8926491.

Bibliography

- [1] Thomas Roelandt, Frank Naets Thijs Willems, and Dirk Vandepitte. High-speed camera measurement of insect-like flapping wing deformation using a speckle pattern wing membrane. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 12–19, Aachen, Germany, Sep 2023. Paper no. IMAV2023-1. 8
- [2] Sang-Gil Lee, Hyeon-Ho Yang, Hwankyun You, and Jae-Hung Han. Design and path following flight test of a flapping-wing air vehicle. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 20–25, Aachen, Germany, Sep 2023. Paper no. IMAV2023-2. 8
- [3] Hyeon-Ho Yang, Sang-Gil Lee, Eun-Hyuck Lee, and Jae-Hung Han. Integrated simulation program for initial design of bird-inspired ornithopter. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 26–32, Aachen, Germany, Sep 2023. Paper no. IMAV2023-3. 8
- [4] Carlos Osorio Quero, Daniel Durini, Jose Rangel-Magdaleno, Jose Martinez-Carranza, and Ruben Ramos-Garcia. 3d human pose reconstruction single-pixel imaging. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 33–39, Aachen, Germany, Sep 2023. Paper no. IMAV2023-4. 8
- [5] Aldrich Alfredo Cabrera-Ponce, Manuel Isidro Martin-Ortiz, and Jose Martinez-Carranza. Hierarchical continual learning for single image aerial localisation. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 40–48, Aachen, Germany, Sep 2023. Paper no. IMAV2023-5. 8
- [6] David Yang, Karl Stol, and Salim Al-Zubaidi. Determining the moment of inertia tensor of a uav using a motion capture system. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 49–56, Aachen, Germany, Sep 2023. Paper no. IMAV2023-6. 8
- [7] Jose Martinez-Carranza, Delia Irazu Hernandez-Farias, Leticia Oyuki Rojas-Perez, and Aldrich Alfredo Cabrera-Ponce. Why do i need to speak to my drone? In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 57–64, Aachen, Germany, Sep 2023. Paper no. IMAV2023-7. 8
- [8] Yingfu Xu and Guido C.H.E. de Croon. A lightweight learning-based visual-inertial odometry. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 65–72, Aachen, Germany, Sep 2023. Paper no. IMAV2023-8. 8
- [9] Ruben Meester, Tom van Dijk, Christophe De Wagter, and Guido C.H.E. de Croon. Frustumbug: a 3d mapless stereo-vision-based bug algorithm for micro air vehicles. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 73–85, Aachen, Germany, Sep 2023. Paper no. IMAV2023-9. 8
- [10] Sven Pfeiffer, Christophe De Wagter, and Guido C.H.E. de Croon. Improved moving horizon estimation for ultra-wideband localization on small drones. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 86–92, Aachen, Germany, Sep 2023. Paper no. IMAV2023-10. 8
- [11] Yingfu Xu and Guido C.H.E. de Croon. Efficient model-aided visual-inertial ego-motion estimation for multirotor mavs. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 93–100, Aachen, Germany, Sep 2023. Paper no. IMAV2023-11. 8
- [12] Iris Liu, Suvarna Swaraj, Karl Stol, and Salim Al-Zubaidi. Analysis of admittance control for a fully-actuated, tactile uav. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 101–108, Aachen, Germany, Sep 2023. Paper no. IMAV2023-12. 8
- [13] Alessandro Mancinelli, Erik Van Der Horst, Bart D.W. Remes, and Ewoud J.J. Smeur. Autopilot framework with indi rpm control, real-time actuator feedback, and stability control on companion computer through matlab generated functions. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 109–116, Aachen, Germany, Sep 2023. Paper no. IMAV2023-13. 8

- [14] Julian Mller, Daniel Schatten, Dominik Felix Duda, and Dieter Moormann. Cascaded indoor flight controller design for a miniaturized tiltwing aircraft. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 117–126, Aachen, Germany, Sep 2023. Paper no. IMAV2023-14. 8
- [15] Caleb Probine and Karl Stol. Performance comparisons on fully-actuated multirotors with different actuation modes. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 127–134, Aachen, Germany, Sep 2023. Paper no. IMAV2023-15. 8
- [16] Bilal Mubdir and Emmanuel Prempain. Distributed nonlinear model predictive control for a quadrotor uav. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 135–142, Aachen, Germany, Sep 2023. Paper no. IMAV2023-16. 8
- [17] Thomas Pavot, Renaud Kiefer, Tedjani Mesbahi, and Edouard Laroche. Adaptive frequency control strategy for the propulsion management of a long-endurance drone using hybrid power sources. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 143–148, Aachen, Germany, Sep 2023. Paper no. IMAV2023-17. 8
- [18] Caleb Probine, David Yang, Karl Stol, and Nicholas Kay. Model predictive control on a fully-actuated octocopter for wind disturbance rejection. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 149–156, Aachen, Germany, Sep 2023. Paper no. IMAV2023-18. 8
- [19] Joshua Taylor, Matthew Edwards, Karl Stol, and Salim Al-Zubaidi. Assessing the agility of a variable-tilt uav. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 157–164, Aachen, Germany, Sep 2023. Paper no. IMAV2023-19. 8
- [20] Salim Al-Zubaidi and Karl Stol. Analysis of the transient response of uav rotors. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 165–170, Aachen, Germany, Sep 2023. Paper no. IMAV2023-20. 8
- [21] Luiz Fernando Tiberio Fernandez, Murat Bronz, Nathalie Bartoli, and Thierry Lefebvre. Multi-vehicle simulation framework for heterogeneous unconventional mavs. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 171–178, Aachen, Germany, Sep 2023. Paper no. IMAV2023-21. 8
- [22] Zakeye Azaki, Pierre Susbielle, Alexis Offermann, Jonathan Dumon, Nacim Meslem, Ahmad Hably, and Amaury Negre. Experimentally validated simulator of flight dynamics: for a magnus effect-based quadcopter system. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 179–186, Aachen, Germany, Sep 2023. Paper no. IMAV2023-22. 8
- [23] Zhen Liu and Wenhao Xie. Numerical investigation on flow control of leading-edge reinforced flexible membrane wing. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 187–192, Aachen, Germany, Sep 2023. Paper no. IMAV2023-23. 8
- [24] Junyi Chen and Karl Stol. Measuring disturbance response of a fully-actuated uav using surging flow wind tunnel experiments. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 193–199, Aachen, Germany, Sep 2023. Paper no. IMAV2023-24. 8
- [25] Lennart Danielmeier, Sebastian Seitz, Max Hartmann, Nicolai Voget, and Dieter Moormann. Multi method local path planning considering flight dynamics constraints for collision avoidance. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 200–207, Aachen, Germany, Sep 2023. Paper no. IMAV2023-25. 8
- [26] Matteo Fei, Andrei Muresan Radu, Wonhee Lee, Marcos Linares Bayon, and Leandro Lustosa Ribeiro. Autonomous landing, obstacle avoidance and block recovery for a quadrotor drone. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 208–215, Aachen, Germany, Sep 2023. Paper no. IMAV2023-26. 8
- [27] Alexander Quessy, Thomas Richardson, and Sebastian East. Automating fixed wing forced landings with offline reinforcement learning. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 216–223, Aachen, Germany, Sep 2023. Paper no. IMAV2023-27. 8
- [28] Mohamad Hachem, Murat Bronz, Thierry Miquel, and Clément Roos. Trajectory optimization for fully actuated hexacopters: Enhancing maneuverability and applications. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 224–230, Aachen, Germany, Sep 2023. Paper no. IMAV2023-28. 8

- [29] Jonathas Laffita van den Hove, Ewoud J.J. Smeur, and Bart D.W. Remes. Rigid airborne docking between a fixed-wing uav and an over-actuated multicopter. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 231–239, Aachen, Germany, Sep 2023. Paper no. IMAV2023-29. [8](#)
- [30] Anton Sebastian Schwager, Philipp Müller, Florian Knaak, and Dieter Moormann. Task allocation for heterogeneous unmanned aerial vehicles. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 240–246, Aachen, Germany, Sep 2023. Paper no. IMAV2023-30. [8](#)
- [31] Yuyang Shen, Fangguo Zhao, Jiahao Mei, Jin Zhou, Jinming Xu, and Shuo Li. Time-optimal trajectory generation with input boundaries and dynamic waypoints constraints for a swarm of quadrotors. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 247–254, Aachen, Germany, Sep 2023. Paper no. IMAV2023-31. [8](#)
- [32] Vayam Jain, Aman Virmani, Nilesch Aggarwal, Anup Kumar Malhotra, Anunay Anunay, and Sarthak Singh. Mtso: Multi-target search optimisation based on probability map. In D. Moormann, editor, *14th annual International Micro Air Vehicle Conference and Competition*, pages 255–262, Aachen, Germany, Sep 2023. Paper no. IMAV2023-32. [8](#)