

http://www.imavs.org/

IMAV 2026

17th International Micro Air Vehicle Conference and Competition

September 21-25, 2026

Strasbourg, France

www.imavs.org

Contents

IMAV2026 1
Preface . . . 3
International IMAV Committee . . . 4
Organizing Committee . . . 4
Reviewers . . . 4
Sponsors . . . 5
Call for Papers . . . 6
Sessions . . . 7
Keynote Speakers . . . 8
List Of Papers . . . 11
Best Paper Awards . . . 12
Authors . . . 13

Papers 15
Rapid Sim-to-Real Reinforcement Learning for Autonomous Drone Racing . . . 16
Precision Home Landing for Bee-Nav Robot Navigation . . . 24
VLN on the Fly: An Onboard Vision-Language Navigation Stack for Aerial Robots . . . 32
MRAC-Based Admittance Control for Aerial Physical Interaction . . . 39
Variable Admittance Control for Obstacle Separation in Aerial Manipulation . . . 47
Online Wind Field Estimation for Airborne Wind Energy Systems in Circular Flight . . . 55
Sequential Estimation of Fixed-Wing UAV Aerodynamic Coefficients: Evaluation on Pitch/Roll Flight Data . . . 63
TriMorph: The Transformative Tricopter . . . 71
Weitflug: A Research-Oriented Fork of Betaflight for Agile Autonomous Flight . . . 78
Flight-Ready LiDAR-Inertial Odometry for Embedded Drone Platforms . . . 85
LiDARFlow: Real-Time Panel-Based MAV Guidance in Unknown Environments . . . 92
Monocular Geo-Localization of Terrestrial Objects from UAV Imagery with Spatial Refinement . . . 100
Actuator Nonlinear Fault-Tolerant Control of a Fast-Spinning Quadrotor with Loss of Two Opposing Rotors . . . 108
AERO-LQG: Aerial-Enabled Robust Optimization for LQG-Based Quadrotor Flight Controller . . . 118
Parameter Sensitivity in Control Allocation: A Monte Carlo Analysis for an Over-Actuated Aerial Robot . . . 126
A Multi-View, Multi-LiDAR Synchronised Setup for Smoke Plume Data Acquisition: A Simulated Drone Swarm Observation Case Study . . . 134
Vision-guided Velocity Control of Micro Air Vehicles using Rate-decoded Spiking Neural Network . . . 142
Decentralized PSO-Based Guidance for Multi-UAV Wildfire Monitoring and Frontier Tracking . . . 150
Distributed Formation Control based on Bounded Voronoi and Cable Tension for Cooperative Hovering of a Cable-Suspended Load . . . 158
A Structure-Aware Reactive Motion Planner for Autonomous Nano-UAV Navigation in Confined 3D Tunnels Using Minimal Range Sensing . . . 166
Downscaling Autonomous Navigation of Tiny Drones under Dense Forest Canopies . . . 173
Safe Synchronous Path Planning for a Fleet of Fixed-Wing Constant Speed Aircraft . . . 181
Center-of-Gravity Placement Effects on a Bio-Inspired Flapping-Wing MAV: An Empirical Characterization for Sensor Payload Integration . . . 189
Pitch Stabilisation and Longitudinal Control of a Hummingbird-Inspired Drone Using an Active Tail . . . 195
Experimental Identification of the Yaw Damping Derivative of a Flapping-Wing Micro Air Vehicle . . . 203

http://www.imavs.org/

Preface

On behalf of the Local Organizing Committee, it is our pleasure to present the proceedings of the 17th International Micro Air Vehicle Conference and Competition, which was held in Strasbourg, France from September 21-25, 2026.
This year, IMAV was supported by INSA Strasbourg and the ICube laboratory. Organized in coordination with the SIS67 fire department and the 2nd Hussard Regiment, this edition focused on managing rescue operations in fire conditions. We extend our warm thanks to all sponsors and organizers for making the event possible.

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 twenty-five peer-reviewed scientific papers by eighty-seven authors organized in eight sessions presented at the IMAV in 2026. 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.

We would like to express our sincere gratitude for the guidance and support of the nineteen members of the International Committee from fifteen institutes, who guide IMAV overall. 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 the five members of the Local Organizing Committee for their invaluable support in the organization of this IMAV 2026.

Strasbourg, France. September 2026

Renaud Kiefer and Sylvain Durand
General Chair and Conference Chair
INSA Strasbourg & ICube
email: renaud.kiefer@insa-strasbourg.fr, sylvain.durand@insa-strasbourg.fr

http://www.imavs.org/

International IMAV Committee

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

Organizing Committee

Renaud Kiefer (INSA Strasbourg) . . . . . General Chair
Sylvain Durand (INSA Strasbourg) . . . . . Conference Chair
Caio Fonseca (INSA Strasbourg) Indoor Competition Chair
Thomas Pavot (INSA Strasbourg) . . . Outdoor Competition Chair
Edouard Laroche (Strasbourg University) . Sponsors Chair

Reviewers

Abuali Saleh . . . . . .
Al-Zubaidi Salim . . . . . .
Alkimim Dos Santos Lívia . . . . . .
Arias-Perez Pedro . . . . . .
Arnous Asil . . . . . .
Asignacion Abner . . . . . .
Bara Iulia . . . . . .
Batista Bruno . . . . . .
Borne Augustin . . . . . .
Chakraa Hamza . . . . . .
Chan Woei-Leong . . . . . .
Changey Sebastien . . . . . .
De Croon Guido . . . . . .
De Ponti Tomaso . . . . . .
De Wagter Christophe . . . . . .
Durand Sylvain . . . . . .
Engelsman Daniel . . . . . .
Fernandez Cortizas Miguel . . . . . .
Feurgard Maël . . . . . .
Fonseca De Freitas Caio . . . . . .
Gangloff Jacques . . . . . .
Gaona Alvaro . . . . . .
Garreau Aurélien . . . . . .
Gonzalez Silva Eduardo . . . . . .
Gp-Lenza Guillermo . . . . . .
Guerrero-Castellanos Fermi . . . . . .
Hafez Ishaq . . . . . .
Harzer Jakob . . . . . .
Hattenberger Gautier . . . . . .
Kempf Aurélie . . . . . .
Klimont Joel . . . . . .
Labhade Jayant Kailas . . . . . .
Laroche Edouard . . . . . .
Lebreton Clément . . . . . .
Machado Joao . . . . . .
Melis Arno . . . . . .
Melo Joao . . . . . .
Mendoza-Robles Natalio . . . . . .
Moschetta Jean-Marc . . . . . .
Ou Dequan . . . . . .
Pavot Thomas . . . . . .
Peng Yufei . . . . . .
Pivert Maya . . . . . .
Pouthier Florian . . . . . .
Pérez Rafael . . . . . .
Rey Thomas . . . . . .
Roussel Emmanuel . . . . . .
Saade Joya Maria . . . . . .
Saeed Burhan . . . . . .
Setayeshi Amirali . . . . . .
Smith Charles . . . . . .
Soprani Tayar Marco . . . . . .
Stol Karl . . . . . .
Viollet Stéphane . . . . . .
Watkins Simon . . . . . .
Willinger-Adam Pierre . . . . . .
Xin Songyan . . . . . .
Xolo-Tlapanco Nilda G. . . . . .
Yigit Arda . . . . . .
Youbi Lounès . . . . . .
Zwanenburg Andreas . . . . . .

http://www.imavs.org/

Organizers

IMAV 2026 was organized by INSA Strasbourg and the ICube laboratory, in coordination with the SIS67 fire department and the 2nd Hussard Regiment. This edition focused on managing rescue operations in fire conditions.

https://www.insa-strasbourg.fr/

https://icube.unistra.fr/

https://www.sis67.alsace/

2e Regiment de Hussards

Sponsors

IMAV 2026 was made possible thanks to the great support of:

https://www.unistra.fr/

https://www.france2030.gouv.fr

https://delair.aero/

https://www.dronisos.com/

http://www.imavs.org/

Call for Papers



About IMAV

The international Micro Air Vehicle Conference and Competition (IMAV) is an annual event combining cutting-edge research and drone competitions. It features keynote talks by renowned experts and peer-reviewed paper presentations.

This year's edition will take place at the Institut National des Sciences Appliquées de Strasbourg (INSA Strasbourg), in Strasbourg, Bas-Rhin, France, on September 21st - 25th, 2026.

IMPORTANT DATES

PAPER SUBMISSION DEADLINE
May 15, 2026

NOTIFICATION OF ACCEPTANCE
June 12, 2026

FINAL PAPER SUBMISSION
July 3, 2026

CONFERENCE TOPICS

- Low Reynolds number aerodynamics
- Unsteady aerodynamics
- Propulsion set and new energy sources
- Smart morphing materials
- Autonomous Drone Racing
- Autonomous navigation
- Control theory and state estimation
- Cooperation and formation flight
- Sense & avoid integration of UAVs in airspace
- Computer vision for MAVs
- Reconfiguration in unpredicted events
- New MAV architectures
- Characterization of noise emission for MAVs
- Low noise and noise mitigation

For all inquiries, please contact: imav2026@insa-strasbourg.fr

Organised by:

INSA Institut National des Sciences Appliquées Strasbourg

iCUBE

Université de Strasbourg

In coordination with:

Sapeurs-Pompiers du Bas-Rhin

Website

2026.imavs.org

http://www.imavs.org/

Conference Oral Presentations

Session 1: Learning and Adaptive Systems

Rapid Sim-to-Real Reinforcement Learning for Autonomous Drone Racing p.16
Precision Home Landing for Bee-Nav Robot Navigation p.24
VLN on the Fly: An Onboard Vision-Language Navigation Stack for Aerial Robots p.32

Session 2: Admittance Control and Aerial Interaction

MRAC-Based Admittance Control for Aerial Physical Interaction p.39
Variable Admittance Control for Obstacle Separation in Aerial Manipulation p.47
Online Wind Field Estimation for Airborne Wind Energy Systems in Circular Flight p.55
Sequential Estimation of Fixed-Wing UAV Aerodynamic Coefficients: Evaluation on Pitch/Roll Flight Data p.63

Session 3: Innovative Architectures and Software Tools

TriMorph: The Transformative Tricopter p.71
Weitflug: A Research-Oriented Fork of Betaflight for Agile Autonomous Flight p.78

Session 4: Advanced LiDAR and Vision for Autonomous Navigation

Flight-Ready LiDAR-Inertial Odometry for Embedded Drone Platforms p.85
LiDARFlow: Real-Time Panel-Based MAV Guidance in Unknown Environments p.92
Monocular Geo-Localization of Terrestrial Objects from UAV Imagery with Spatial Refinement p.100

Session 5: Robust Control and Fault Tolerance

Actuator Nonlinear Fault-Tolerant Control of a Fast-Spinning Quadrotor with Loss of Two Opposing Rotors p.108
AERO-LQG: Aerial-Enabled Robust Optimization for LQG-Based Quadrotor Flight Controller p.118
Parameter Sensitivity in Control Allocation: A Monte Carlo Analysis for an Over-Actuated Aerial Robot p.126

Session 6: Swarm Guidance and Multi-Sensor Perception

A Multi-View, Multi-LiDAR Synchronised Setup for Smoke Plume Data Acquisition: A Simulated Drone Swarm Observation Case Study p.134
Vision-guided Velocity Control of Micro Air Vehicles using Rate-decoded Spiking Neural Network p.142
Decentralized PSO-Based Guidance for Multi-UAV Wildfire Monitoring and Frontier Tracking p.150
Distributed Formation Control based on Bounded Voronoi and Cable Tension for Cooperative Hovering of a Cable-Suspended Load p.158

Session 7: Navigation in Complex Environments

A Structure-Aware Reactive Motion Planner for Autonomous Nano-UAV Navigation in Confined 3D Tunnels Using Minimal Range Sensing p.166
Downscaling Autonomous Navigation of Tiny Drones under Dense Forest Canopies p.173
Safe Synchronous Path Planning for a Fleet of Fixed-Wing Constant Speed Aircraft p.181

Session 8: Bio-Inspired MAVs: Design and Control

Center-of-Gravity Placement Effects on a Bio-Inspired Flapping-Wing MAV: An Empirical Characterization for Sensor Payload Integration p.189
Pitch Stabilisation and Longitudinal Control of a Hummingbird-Inspired Drone Using an Active Tail p.195
Experimental Identification of the Yaw Damping Derivative of a Flapping-Wing Micro Air Vehicle p.203

http://www.imavs.org/

Keynote Speakers

Aerial Robotic Cable-Based Manipulation: Advancements and Emergent Challenges

Prof. Chiara Gabellieri – Assistant Professor in the Robotics and Mechatronics group of the EEMCS Faculty at the University of Twente, the Netherlands

Abstract

New research directions are emerging in both aerial manipulation of cable-suspended objects and the control of deformable cables with aerial robots, calling for novel modeling and control approaches. Beyond the well-studied problem of controlling cable-suspended loads with multirotor UAVs, recent work—primarily motivated by endurance limitations—is exploring manipulation using non-stop aerial vehicles. This paradigm introduces the challenge of achieving accurate control of the load pose while maintaining continuous motion of the carriers. Moreover, traditional cable models become insufficient when highly dynamic tasks are considered or where the cable itself becomes the deformable object to be shaped and controlled. Developments in this direction enables applications ranging from highly dynamic pick-and-place of slung loads to the collection of floating debris in waterways.



Counter-UAS Challenges and Research Perspectives: Building Sustainable and Adaptive Defense Architectures

Laurent Barrilliet – Program Director at ONERA[1], Versailles, France

Abstract

The rapid proliferation of low-cost, easily accessible drones has transformed the security and defense landscape. Recent conflicts have demonstrated how unmanned systems can provide air superiority and long-range strike capabilities at a fraction of the cost of conventional military assets. This asymmetry creates major challenges for Counter-UAS (C-UAS) systems, which must remain effective, sustainable, and scalable against evolving threats.

Designing efficient C-UAS architectures requires a comprehensive understanding of drone technologies, their potential evolution, and their impact on detection, information management, and neutralization capabilities. These architectures must also operate in contested environments characterized by GNSS denial, electronic warfare, operational constraints, and regulatory requirements.

Drawing on ONERA's expertise, instrumented drone fleet, test facilities, and multidisciplinary research laboratories, this presentation reviews key drone threat trends, associated risks, and ongoing research efforts aimed at reducing vulnerabilities. Operational examples, including the DIAMOND drone-track fusion algorithm and the SAP system deployed during the Paris 2024 Olympic Games, illustrate the transition from research to fielded capabilities.



[1]The French Aerospace Lab ONERA is a leading research organization dedicated to aerospace, defense, and space technologies

http://www.imavs.org/

Centimeter-level precision for a large-scale pointillist artwork created by a drone (a world first)

Prof. Jacques Gangloff – Professor at Strasbourg University, France

Abstract

The highlight of the ANR "STRAD" (STReet Art Drone) project was the autonomous creation in July 2025 of a 5 m x 15 m outdoor mural by an aerial manipulator with centimeter-level precision. A world first. This achievement is the result of a four-year collaboration by a unique consortium comprising the ICube laboratory in Strasbourg, the Gipsa-lab in Grenoble, the company Polyvionics in Paris, and the Spacejunk association in Grenoble, which organizes Europe's largest street art festival, the "Street Art Fest Grenoble Alpes." This presentation will tell the story of this unique project at the intersection of science and art, from its inception to the completion of the final mural as part of the Street Art Fest 2025.



http://www.imavs.org/

The quest for never-landing drones. From hydrogen to wind gradients

Prof. Jean-Marc Moschetta – Professor of Aerodynamics at ISAE-SUPAERO, Toulouse, France

Abstract

Fixed-wing drones powered by an electric motor can only achieve limited flight times due to the intrinsically low energy density of current Lithium batteries. On-board hydrogen can dramatically extend flight times if properly combined with efficiently designed fuel cells. Moreover, the use of cryogenic hydrogen can allow for even higher energy density if the airplane's consumption is properly adapted to the tank's natural boil-off rate. While crossing the Atlantic Ocean (3000 km) with an electric fixed-wing aircraft equipped with Lithium cells is impossible at any take-off weight, that task is feasible for a liquid hydrogen-powered drone of less than 15 kg. In addition to that, taking advantage of wind gradients could further extend the maximum range of drones with the objective of staying airborne for several days. The presentation will finally discuss the use of a Lidar Doppler in view of take advantage of atmospheric disturbances.



Building long-range aerial sensors (Defense and industrial applications, since 2011)

Bastien Mancini – CEO of DELAIR[2], Toulouse, France

Abstract

Delair was founded in 2011 by Michaël de Lagarde, Benjamin Benharrosh, Benjamin Michel, and Bastien Mancini, and quickly became a pioneer in professional UAV systems. In 2012, its DT18 became the first drone certified in France for BVLOS (Beyond Visual Line of Sight) operations, marking a major milestone for the company and the industry.

Over the years, Delair expanded through strategic acquisitions and partnerships, including Airware (and its subsidiary Redbird), Gatewing from Trimble, and collaborations with Intel and international distributors. The company also supplied drones for defense and surveillance applications, notably in Ukraine, and strengthened its presence in both civilian and military markets.

More recently, Delair has become a key actor in European defense drone programs, working with partners such as KNDS and MBDA on loitering munitions and tactical UAVs. Its systems, including the DT-46 and new hybrid platforms, are now used by French armed forces and other security actors, reflecting its shift toward high-performance dual-use technologies and autonomous systems.



http://www.imavs.org/

Implementing Ultra-Wideband (UWB) Technology in Robot Swarms

Rémi Potet & Antoine Potet – Chief Technology Officers of Dronisos[3], Bordeaux, France

Abstract

This presentation examines the integration of Ultra-Wideband (UWB) technology within robot swarms, outlining its operational capabilities and the technical phases of implementation. The keynote will address the following key areas:

- Dynamic Infrastructure: An analysis of the operational viability and implications of using moving or flying anchors instead of static reference points.
- UWB Capabilities: An overview of UWB performance specifications for multi-agent coordination.
- Positioning Techniques: An evaluation of standard UWB localization architectures, specifically RTDOA, TDOA, and TWR.
- Technical Challenges: A review of obstacles in real-world deployment, focusing on clock synchronization, multipath mitigation, time-slot allocation, and error characterization in sensor fusion.



[2]French company that designs and manufactures professional fixed-wing drones and data solutions for industrial inspection, surveying, and defense applications

[3]French company specializing in autonomous drone light shows, developing swarm technologies and large-scale aerial entertainment solutions for events worldwide

Papers

[1] Rapid Sim-to-Real Reinforcement Learning for Autonomous Drone Racing p.16
[2] Precision Home Landing for Bee-Nav Robot Navigation p.24
[3] VLN on the Fly: An Onboard Vision-Language Navigation Stack for Aerial Robots p.32
[4] MRAC-Based Admittance Control for Aerial Physical Interaction p.39
[5] Variable Admittance Control for Obstacle Separation in Aerial Manipulation p.47
[6] Online Wind Field Estimation for Airborne Wind Energy Systems in Circular Flight p.55
[7] Sequential Estimation of Fixed-Wing UAV Aerodynamic Coefficients: Evaluation on Pitch/Roll Flight Data p.63
[8] TriMorph: The Transformative Tricopter p.71
[9] Weitflug: A Research-Oriented Fork of Betaflight for Agile Autonomous Flight p.78
[10] Flight-Ready LiDAR-Inertial Odometry for Embedded Drone Platforms p.85
[11] LiDARFlow: Real-Time Panel-Based MAV Guidance in Unknown Environments p.92
[12] Monocular Geo-Localization of Terrestrial Objects from UAV Imagery with Spatial Refinement p.100
[13] Actuator Nonlinear Fault-Tolerant Control of a Fast-Spinning Quadrotor with Loss of Two Opposing Rotors . . . p.108
[14] AERO-LQG: Aerial-Enabled Robust Optimization for LQG-Based Quadrotor Flight Controller p.118
[15] Parameter Sensitivity in Control Allocation: A Monte Carlo Analysis for an Over-Actuated Aerial Robot p.126
[16] A Multi-View, Multi-LiDAR Synchronised Setup for Smoke Plume Data Acquisition: A Simulated Drone Swarm Observation Case Study p.134
[17] Vision-guided Velocity Control of Micro Air Vehicles using Rate-decoded Spiking Neural Network p.142
[18] Decentralized PSO-Based Guidance for Multi-UAV Wildfire Monitoring and Frontier Tracking p.150
[19] Distributed Formation Control based on Bounded Voronoi and Cable Tension for Cooperative Hovering of a Cable-Suspended Load p.158
[20] A Structure-Aware Reactive Motion Planner for Autonomous Nano-UAV Navigation in Confined 3D Tunnels Using Minimal Range Sensing p.166
[21] Downscaling Autonomous Navigation of Tiny Drones under Dense Forest Canopies p.173
[22] Safe Synchronous Path Planning for a Fleet of Fixed-Wing Constant Speed Aircraft p.181
[23] Center-of-Gravity Placement Effects on a Bio-Inspired Flapping-Wing MAV: An Empirical Characterization for Sensor Payload Integration p.189
[24] Pitch Stabilisation and Longitudinal Control of a Hummingbird-Inspired Drone Using an Active Tail p.195
[25] Experimental Identification of the Yaw Damping Derivative of a Flapping-Wing Micro Air Vehicle p.203

http://www.imavs.org/

Best Paper Awards and Special issue

IMAV 2026 awarded best papers in two categories: "*Methods & Algorithms*" and "*Systems & Applications*", based on the reviewers' feedback and among a list of relevant, well-written, and well-presented publications. This list below of finalists for best paper awards is also a shortlist of papers to be published in a Special Issue of Unmanned Systems.

Finalists for the Best Paper Award in Methods & Algorithms

The selected papers are:

[5] Pedro H.M. Souza, Karl Stol, and Ishaq Hafez (Univ. Auckland, New Zealand), "*Variable Admittance Control for Obstacle Separation in Aerial Manipulation*" p.47
[6] Jakob Harzer, Felix Renard, Kim Polkläser, Dieter Moormann, and Moritz Diehl (Univ. Freiburg & Aachen, Germany), "*Online Wind Field Estimation for Airborne Wind Energy Systems in Circular Flight*" p.55
[13] Tomaso M.L. de Ponti, Ewoud J.J. Smeur, and Bart W.D. Remes (TU Delft, The Netherlands), "*Actuator Nonlinear Fault-Tolerant Control of a Fast-Spinning Quadrotor with Loss of Two Opposing Rotors*" p.108

The **Best Paper Award** for IMAV 2026 in the category "*Methods & Algorithms*" is:

[13] Tomaso M.L. de Ponti, Ewoud J.J. Smeur, and Bart W.D. Remes (TU Delft, The Netherlands), "*Actuator Nonlinear Fault-Tolerant Control of a Fast-Spinning Quadrotor with Loss of Two Opposing Rotors*" p.108

Finalists for the Best Paper Award in Systems & Applications

The selected papers are:

[2] Dequan Ou, Quentin Missinne, Christophe De Wagter, and Guido C.H.E. de Croon (TU Delft, The Netherlands), "*Precision Home Landing for Bee-Nav Robot Navigation*" p.24
[9] Joel Klimont, Jakob Buchsteiner, Alexander Lampalzer, and Radu Grosu (TU Vienna, Austria), "*Weitflug: A Research-Oriented Fork of Betaflight for Agile Autonomous Flight*" p.78
[20] Abner Asignacion Jr, and Satoshi Suzuki (Univ. Chiba, Japan), "*A Structure-Aware Reactive Motion Planner for Autonomous Nano-UAV Navigation in Confined 3D Tunnels using Minimal Range Sensing*" p.166
[21] Andreas Zwanenburg, Salua Hamaza (TU Delft, The Netherlands), "*Downscaling Autonomous Navigation of Tiny Drones under Dense Forest Canopies*" p.173

The **Best Paper Award** for IMAV 2026 in the category "*Systems & Applications*" is:

[9] Joel Klimont, Jakob Buchsteiner, Alexander Lampalzer, and Radu Grosu (TU Vienna, Austria), "*Weitflug: A Research-Oriented Fork of Betaflight for Agile Autonomous Flight*" p.78

http://www.imavs.org/

Authors

A

Abuali, Saleh p.158
Al-zubaidi, Salim p.126
Asignacion Jr, Abner p.166

B

Barros, Edna p.100
Becker, Marcelo p.32
Bello, Ivan Guerra p.189
Bilgin, Zeynep p.92
Briñón-Arranz, Lara p.158
Bronz, Murat p.92, p.134, p.150
Buchsteiner, Jakob p.16, p.78

C

Campoy, Francisco Anguita,Pascual p.85
Capezzuto, Gianluca p.32
Chakraa, Hamza p.134, p.150
Chan, Woei-Leong p.203
Croon, Guido de p.24
César, Pedro p.100

D

Delmas, Damien p.63
Denis, Lou p.134
Diehl, Moritz p.55
Durand, Nicolas p.181
Durand, Sylvain p.158

E

Eduardo, Bruno p.100
Engelsman, Daniel p.118

F

Farnault, Jules p.63
Ferreira, Mateus p.100
Feurgard, Maël p.150, p.181
Freitas, Pedro de p.32

G

Gallo, Mauro p.189
Gaona, Alvaro p.85
Godoy, Ricardo p.32
Grosu, Radu p.16, p.78

H

Hafez, Ishaq p.39, p.47
Hamaza, Salua p.173
Harzer, Jakob p.55
Hattenberger, Gautier p.181
Ho, Hann Woei p.142

I

Ingargiola, Jean-Marc p.71

K

Kempf, Aurélie p.63
Kido, Lucas p.32
Klein, Itzik p.118
Klimont, Joel p.16, p.78

L

Lacroix, Simon p.181
Lampalzer, Alexander p.16, p.78
Laroche, Edouard p.63
Lee, Jasper p.71
Li, Ni p.142
Liu, De-Jing p.203
Lopez, Austin p.189

M

Machado, João p.92
Mafalado, João Pedro p.100
Marchand, Nicolas p.158
Melis, Arno p.195
Melo, João p.100
Mendoza-Robles, Natalio p.158
Missinne, Quentin p.24
Moormann, Dieter p.55

O

Ou, Dequan p.24

http://www.imavs.org/

P

Peng, Yufei p.142
Perez-Saura, David p.85
Pires, Driele p.100
Polkläser, Kim p.55
Ponti, Tomaso De p.108
Pérez, Gaétan p.134

R

Remes, Bart p.108
Renard, Felix p.55
Riquelme, Heitor p.100
Robu, Bogdan p.158
Roelandt, Thomas p.195
Rosa, Paulo p.100
Roussel, Emmanuel p.63

S

Saraiva, Pedro Antinio Rabelo p.32
Skinner, Jaap p.126
Smeur, Ewoud p.108
Sonego, Guilherme p.32
Souza, Pedro p.47
Stol, Karl p.39, p.47, p.126
Strub, Guillaume p.63
Suzuki, Satoshi p.166
Swevers, Jan p.195

T

Tayar, Marco p.32
Tommaselli, Felipe p.32

V

Vandepitte, Dirk p.195
Vanpoucke, Jeroen p.195
Verdoucq, Matthieu p.92
Viollet, Stéphane p.71

W

Wagter, Christophe De p.24

Z

Zwanenburg, Andreas p.173

http://www.imavs.org/

http://www.imavs.org/

Release

V1.0, Rev.16816, 2026-10-17 15:30
Created by: Renaud Kiefer and Sylvain Durand and C. De Wagter

Rapid Sim-to-Real Reinforcement Learning for Autonomous Drone Racing

Joel Klimont*, Jakob Buchsteiner, Alexander Lampalzer and Radu Grosu
Technical University of Vienna, Austria

http://www.imavs.org/

ABSTRACT

Fast sim-to-real iteration is important in autonomous drone racing, where each new track or vehicle requires adaptation under a strict wall-clock budget. We present a JAX-based reinforcement-learning pipeline that trains compact feedforward policies in a massively parallel environment. We identify a compact control-oriented acro-mode quadrotor model from short flight logs and randomize its transfer-relevant parameters. The model captures closed-loop body-rate and thrust response, battery-voltage effects, and direction-dependent drag. To suppress bang-bang control, we regularize action changes and large commands. Over a four-second rollout, this reduces the fraction of outputs above 95% of the maximum action magnitude from 50.0% to 1.1% while maintaining comparable track progress. On an NVIDIA GeForce RTX 5070 Ti, the fastest configuration processes 32 million environment steps per second. A 96-configuration study and a 32-seed evaluation show that rollout and optimizer choices strongly affect training reliability within five minutes. Two policies deployed on the A2RL Season 2 competition track reached speeds near $20\,\mathrm{m/s}$. These results demonstrate rapid policy training and real-world transfer using a compact dynamics model.

1 INTRODUCTION

Autonomous drone racing is a demanding benchmark for agile robotics because it combines high-speed motion, tight spatial constraints, limited onboard computation, and little tolerance for control error. The field has progressed from early academic racing challenges [1–4] to real-world systems operating above $20\,\mathrm{m/s}$ [5, 6]. Learning-based controllers have become competitive when simulation, objectives, and transfer assumptions are designed together [6–8].

Competition deployment requires fast adaptation. High-fidelity simulators can reduce model mismatch, but demand detailed models, greater implementation effort, and more computation. Compact control-oriented models can be estimated from short flight logs and simulated at high throughput, provided they retain the dynamics that govern transfer. This trade-off is central to racing, where every new track requires policy retraining and a new vehicle additionally requires model identification.

We identify a compact control-oriented quadrotor model from short flight logs and use it for competition-scale policy training. The policy outputs body-rate and thrust commands through the acro interface used by racing flight controllers. The simulator omits motor-level actuation and moment-based attitude dynamics and instead models the closed-loop response from policy commands to vehicle motion. We train with proximal policy optimization (PPO) [9]; its batched rollout and update structure maps well to a massively parallel JAX implementation [10] and supports stable training under short wall-clock budgets. Because the compact model omits inner-loop dynamics, the objective penalizes rapid action changes and persistently large deterministic commands that would otherwise produce bang-bang behavior at the deployed interface.

The main contribution is a practical sim-to-real workflow that combines short-log model identification, high-throughput policy training, configuration screening under a fixed wall-clock budget, and real-world validation. We evaluate 96 training configurations, repeat three representative settings over 32 seeds, quantify the effect of action regularization, and deploy two policies on the A2RL Season 2 competition track at speeds near $20\,\mathrm{m/s}$. The training environment and reported configurations are publicly available.[1]

2 RELATED WORK

Drone racing stresses perception, estimation, planning, control, and onboard computation simultaneously [2, 4]. Model-based systems remain strong baselines: model predictive contouring control optimizes progress under dynamic constraints [11], while sampling-based methods such as MPPI perform online trajectory optimization through parallel rollouts [12]. Our compact model serves a different purpose: it supports offline policy training and configuration selection, while deployment requires only a feedforward-network evaluation.

Recent reinforcement-learning systems have achieved highly competitive real-world performance. Swift demonstrated champion-level racing with simulation-trained policies and data-driven residual models [7]. Ferede et al. used domain randomization to transfer a motor-command policy

*Email address(es): joel.klimont@tuwien.ac.at

[1] Project repository: `https://github.com/F-WuTS/jax-drone-racing-open`.

http://www.imavs.org/

across substantially different quadrotors [8]. SkyDreamer and MonoRace address high-speed racing under realistic visual sensing constraints [5, 6]. We study a complementary state-based setting in which a deliberately simple simulator enables very high-throughput policy training while retaining sufficient fidelity for real-world deployment.

Direct training-time comparisons are difficult because control interfaces, tasks, hardware, and deployment criteria differ. Swift reports 10^8 environment interactions in approximately 50 min on an RTX 3090, followed by a 2×10^7-interaction fine-tuning stage [7]. Learning to Fly in Seconds reports direct-RPM trajectory-control transfer after 18 s of training on a consumer laptop and a Crazyflie nano quadrotor [13]. Ferede et al. train each racing-policy variant for 10^8 steps across 3-inch and 5-inch platforms [8], but do not report a directly comparable wall-clock time. For broad context, our $(6144, 64, 16, 4)$ configuration uses 6144 environments, 64 rollout steps, 16 minibatches, and four update epochs. It generates more than 5×10^9 environment transitions in five minutes, while the fastest tested configuration reaches 3.2×10^7 transitions per second on an RTX 5070 Ti. These figures are not a direct speed ranking; we instead ask whether a competition-scale gate-racing policy can be trained reliably within five minutes.

The sim-to-real gap remains central at high speed. Errors in thrust, drag, response delay, battery behavior, and state estimation can substantially change closed-loop behavior. Domain randomization addresses these errors by training over a distribution of model and observation parameters [7, 8]. Related identification work highlights thrust curves, aerodynamic drag, and actuator dynamics as important for real flight [14, 15]; we retain these effects in a compact control-oriented form. Continuous-control policies can also converge to saturated or bang-bang commands [16], and policy-smoothness regularization can reduce high-frequency actuation on real robots [17]. We apply this idea directly to the body-rate and thrust interface and evaluate the resulting command signals.

3 PROBLEM STATEMENT

We train a compact feedforward neural policy for high-speed autonomous drone racing. The track is represented as an ordered cyclic sequence of gates

$$\mathcal{G} = \{G_i\}_{i=1}^{N}, \tag{1}$$

where each gate has a pose and aperture radius. For the A2RL track, the sequence also contains two virtual gates near the chicane and ladder. In these sections, the next physical gate alone does not sufficiently constrain the intended route and permits shortcuts or trajectories that reduce visibility of subsequent gates. The virtual gates use the same transition and reward logic as physical gates, but do not represent physical obstacles. A lap is completed after the vehicle traverses the ordered sequence and crosses the first gate again. The objective is to minimize lap time while avoiding missed gates, track-boundary violations, and unsafe flight states.

The learned policy is deliberately track-specific: gate geometry enters both the observation and reward, so a materially different layout requires retraining. The short training budget is intended to make this adaptation practical. We focus on control learning; perception and state estimation provide the track-relative policy state but are outside the scope of this paper.

3.1 Learning Formulation

The real task is partially observable because of estimator uncertainty, actuation uncertainty, latency, battery variation, and unmodeled aerodynamics. During training, we approximate it as a randomized finite-horizon Markov decision process with noisy observations. The simulator state is

$$s_t = (x_t, i_t, h_t, \theta), \tag{2}$$

where x_t is the vehicle state, i_t the target-gate index, h_t recent actions, and θ sampled physical parameters. The policy receives a noisy observation $\tilde{o}_t$, samples an unconstrained Gaussian action, and applies a hyperbolic tangent,

$$\begin{aligned} &\hat{a}_t \sim \pi_\phi(\hat{a}_t \mid \tilde{o}_t), \\ &a_t = \tanh(\hat{a}_t) \in [-1, 1]^4, \end{aligned} \tag{3}$$

where a_t contains normalized roll-rate, pitch-rate, yaw-rate, and thrust commands. PPO optimizes

$$J(\phi) = \mathbb{E}\left[\sum_{t=0}^{H-1} \gamma^t r_t\right] \tag{4}$$

over randomized racing episodes.

4 METHODS

We describe the compact simulator, policy interface, reward design, and randomized PPO training setup. The environment is deliberately simpler than a motor-level quadrotor simulator, but retains the closed-loop response effects selected as most relevant for transfer.

4.1 Compact Quadrotor Model

The simulated state contains position, velocity, attitude, filtered command state, battery voltage, and body rates,

$$x_t = (p_t, v_t, q_t, u_t, V_t, \omega_t). \tag{5}$$

The bounded policy commands $a_t = \tanh(\hat{a}_t) \in [-1, 1]^4$ pass through a first-order response model,

$$u_t = u_{t-1} + \alpha(a_t - u_{t-1}), \qquad \alpha = 1 - \exp\left(-\frac{\Delta t}{\tau}\right), \tag{6}$$

with separate time constants for roll, pitch, yaw, and thrust. Filtered rate commands are scaled by the identified maximum

http://www.imavs.org/

Table 1: Model-identification errors on the chronological validation split. Rate errors are in deg/s and thrust errors in m/s^2.

Response	τ [ms]	RMSE	MAE	R^2
Roll rate	6.7	10.68	7.37	0.994
Pitch rate	12.1	12.94	9.61	0.978
Yaw rate	12.4	6.24	4.33	0.976
Thrust	16.8	0.29	0.16	0.993

body rates and integrated directly into the attitude. The simulator therefore omits individual motors, moment-based attitude dynamics, propeller aerodynamics, and the internal rate controller.

Collective thrust is computed from filtered thrust and battery voltage as

$$f_T = c_0 + c_1 u_{t,\mathrm{thr}} + c_2 u_{t,\mathrm{thr}}^2 + c_3 u_{t,\mathrm{thr}}^3 + c_4 V_t + c_5 u_{t,\mathrm{thr}} V_t. \tag{7}$$

The translational dynamics combine thrust, gravity, and aerodynamic drag,

$$\dot{p}_t = v_t, \tag{8}$$

$$\dot{v}_t = R(q_t)\begin{bmatrix}0 & 0 & f_T/m\end{bmatrix}^\top + \begin{bmatrix}0 & 0 & -g\end{bmatrix}^\top + f_D/m. \tag{9}$$

Drag uses an ellipsoid approximation of the vehicle body, making projected area direction-dependent, and battery voltage follows a linear discharge model. The result is a fast control-oriented approximation whose dominant uncertain parameters can be randomized during training. The exact nominal response, thrust, drag, and body-rate parameters are provided in the project repository.

We identify the model from approximately five minutes of manually flown data spanning the battery-discharge range. After preprocessing, the body-rate dataset contains 40,389 samples and the thrust dataset 339,948 samples. Both use a chronological 70/30 training-validation split, yielding 28,272/12,117 rate samples and 237,963/101,985 thrust samples. For each rate axis, bounded nonlinear least squares estimates the first-order time constant by minimizing sample-wise setpoint-response residuals; a constant offset is computed from the mean residual. For thrust, bounded nonlinear least squares estimates the response time constant, while the six coefficients in Eq. (7) are solved by linear least squares for each candidate time constant to minimize body-frame vertical-acceleration error. Figure 1 compares measured and predicted body-frame vertical acceleration, and Table 1 reports validation errors. Additional validation plots and the exact nominal parameters are provided in the project repository.

4.2 Observation and Action Interface

On the real system, an EKF fuses IMU data and gate detections to estimate the vehicle state in the world frame.

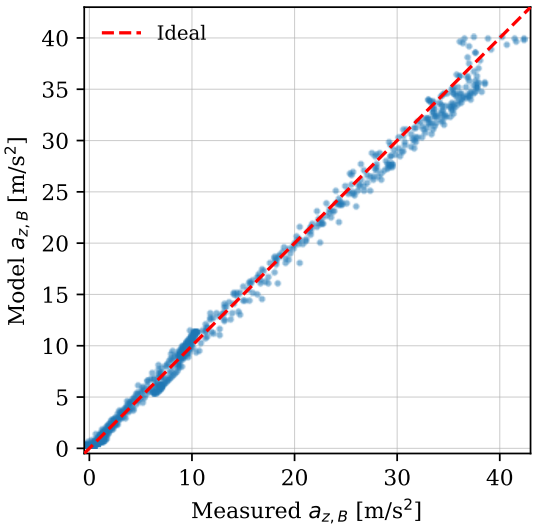
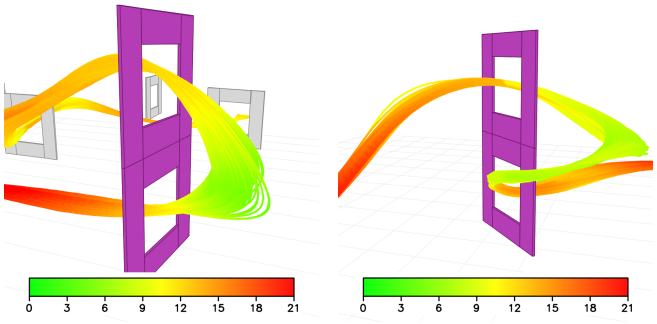


Figure 1: Measured and predicted body-frame vertical acceleration on the chronological validation split.



(a) Split-S (b) Ladder

Figure 2: Complex maneuvers on the A2RL Season 2 track. Color denotes policy speed in meters per second.

The racing environment converts this estimate into current-gate coordinates, keeping the policy input independent of the global track frame while preserving the same feature construction in simulation and deployment. The observation is

$$o_t = [p_t^{G_i}, v_t^{G_i}, \xi_t^{G_i}, \omega_t^{G_i}, u_t, a_{t-1:t-3}, V_t, d_{i,i+1}^{G_i}, d_{i,i+2}^{G_i}], \tag{10}$$

where $p_t^{G_i}$ and $v_t^{G_i}$ denote position and velocity in the current gate frame, $\xi_t^{G_i}$ is relative attitude, $\omega_t^{G_i}$ are gate-frame angular rates, u_t is the filtered command state, $a_{t-1:t-3}$ are the previous three actions, V_t is battery voltage, and the final terms describe the next two gate offsets. Gaussian observation noise approximates estimator uncertainty during training.

4.3 Gate Passing and Termination

Each gate defines a plane and aperture disk. A pass is registered only when the drone crosses the plane through the

http://www.imavs.org/

disk; crossings outside the disk are missed gates. Episodes terminate after a missed gate, flight-volume violation, battery or altitude safety violation, gate timeout, or maximum horizon. We do not add a separate gate-collision penalty beyond termination and the lost opportunity for further progress, since preliminary penalties encouraged overly conservative flight near gates, consistent with Ferede et al. [8].

4.4 Reward and Action Regularization

The task reward combines radial progress toward the active gate, a sparse gate-passing bonus, lap-time pressure, altitude and timeout penalties, and a distance-weighted gate-facing term. The latter discourages the vehicle from pointing away from a nearby gate and improves observability for the onboard EKF by keeping upcoming gates in view.

To discourage control behavior that exploits omitted inner-loop dynamics, we penalize temporal changes in consecutive squashed sampled actions,

$$\tilde{r}_t = r_t - \lambda_{\Delta a} \left\| \tanh(\hat{a}_t) - \tanh(\hat{a}_{t-1}) \right\|_2^2, \qquad (11)$$

and add an ℓ_2 penalty on the squashed deterministic policy mean,

$$\mathcal{L}_{\text{actor}} = \mathcal{L}_{\text{PPO}} + \lambda_a \mathbb{E}_t \left[\left\| \tanh(\mu_\phi(o_t)) \right\|_2^2 \right]. \qquad (12)$$

The first term suppresses rapid command variation; the second discourages unnecessarily large mean commands and persistent saturation. Section 5 evaluates their combined effect on the deployed command interface.

4.5 Policy Optimization and Reset Distribution

We train PPO in the vectorized JAX environment with independently randomized initial states, observations, actions, and dynamics parameters. Actor and critic are feedforward MLPs with two 64-unit $\tanh$ hidden layers. The same architecture is used throughout the configuration study, seed study, and deployment. All reported timing experiments use an NVIDIA GeForce RTX 5070 Ti with 16 GB of VRAM.

Resets are sampled along a cyclic Catmull-Rom spline through the start position and gate centers. Position, speed, direction, and attitude are perturbed, and the target gate is assigned consistently with sampled progress. This broadens state-space coverage without requiring the deployed policy to track the spline. The exact reset and noise parameters are provided in the project repository.

Physical parameters are randomized independently per episode,

$$\theta_{\text{sim}} = \theta_{\text{nom}} \odot \kappa, \qquad \kappa_i \sim \mathcal{U}(\kappa_i^{\min}, \kappa_i^{\max}). \qquad (13)$$

Table 2 lists the randomized parameters. The distribution is intentionally low-dimensional and focuses on effects that strongly influence the closed-loop response. Mass and gravity remain fixed because mass is largely confounded with thrust scale through f_T/m, while gravity is known accurately.

Table 2: Domain-randomized model parameters. Scale factors are sampled independently per episode.

Parameter	Randomization	Description
$\tau_\phi, \tau_\theta, \tau_\psi, \tau_T$	$\mathcal{U}(0.8, 1.2)$	Command-response time constants
$\omega^x_{\max}, \omega^y_{\max}, \omega^z_{\max}$	$\mathcal{U}(0.8, 1.2)$	Maximum body rates
s_T	$\mathcal{U}(0.8, 1.2)$	Thrust-output scale
C_D	$\mathcal{U}(0.8, 1.2)$	Drag scale

5 EVALUATION

We evaluate the workflow in four steps. First, we compare rollout and optimizer settings under the five-minute budget. We then test repeatability across seeds and quantify the effect of action regularization. Finally, we examine real-world transfer and where the compact model remains informative despite its simplifications.

The track includes the split-S and ladder shown in Figure 2. The split-S combines a large heading reversal with vertically separated gate passages, while the ladder requires rapid altitude changes without sacrificing forward speed. We use passage of these elements as intermediate training milestones.

5.1 Five-Minute Training-Configuration Study

Figure 3 summarizes a 96-run study over 12 combinations of parallel-environment count and rollout length. The tested environment counts are 2048, 4096, 6144, 8192, 10240, and 12288; the rollout lengths are 64, 128, 256, and 512 steps. The project repository lists the exact pairings. Each rollout configuration is combined with 8, 16, 32, or 64 minibatches and either four or six optimizer epochs, while the model, network, reward, and nominal five-minute target remain fixed. The complete offline configuration study totals 480 nominal GPU-minutes when run sequentially and is not part of the per-track deployment cost.

The timed interval begins with the first JAX training call and includes compilation, rollout collection, PPO updates, and snapshot capture. Setup occurs beforehand; benchmarks, host logging, checkpoint serialization, and export occur afterward. The fastest configuration reaches 32 million environment steps per second on the RTX 5070 Ti, but does not achieve the highest lap-completion rate. Some lower-throughput configurations learn faster and more reliably within the same budget, showing that rollout length, batch structure, and update allocation matter at least as much as raw simulation speed.

Snapshots are evaluated with fixed-start and random-start benchmarks using 256 agents in the configuration study and 512 in the seed study. Each benchmark includes one nominal model, while the remaining agents independently sample all randomized model parameters; observation and action noise are disabled. Rollouts last at most 1000 steps, or 10 s at 100 Hz. We call a policy deployment-ready when at

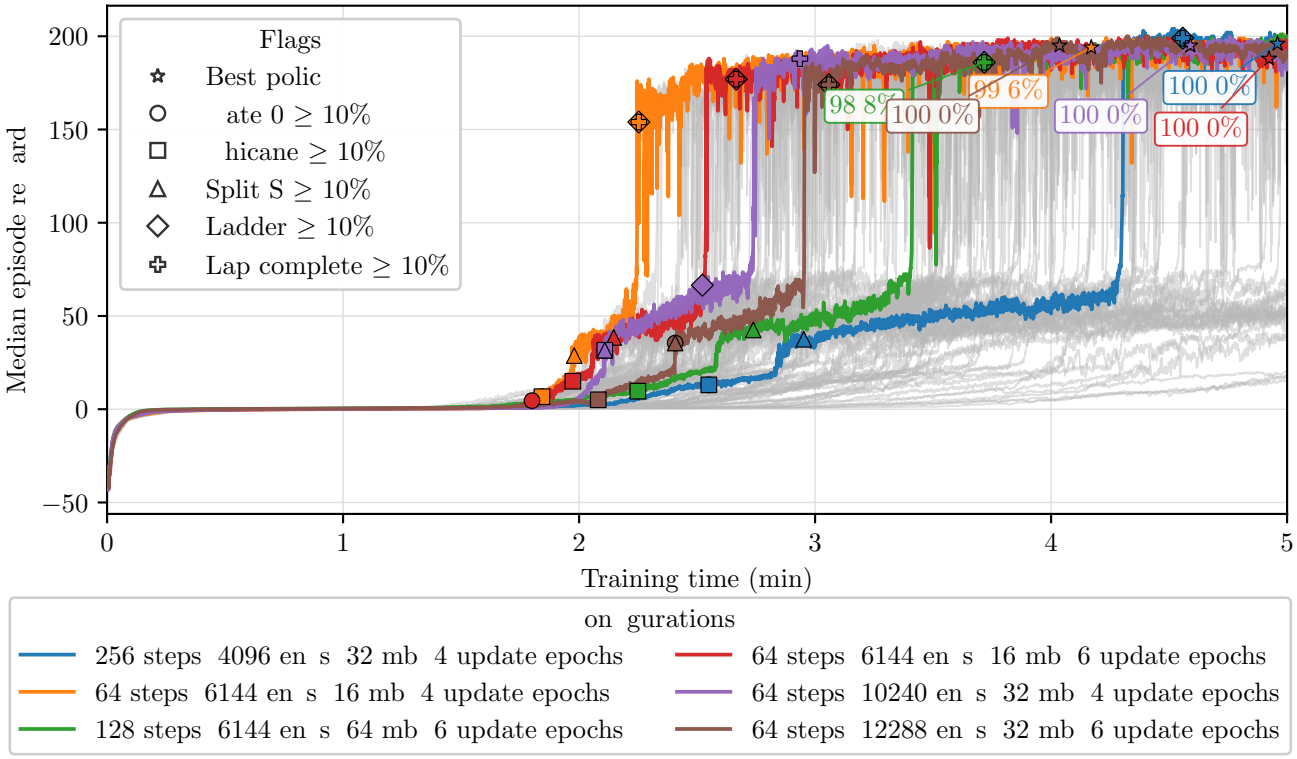


Figure 3: Training-configuration study over 96 PPO settings with a nominal five-minute target per run. Curves show median episode reward. Deferred domain-randomized benchmarks provide the track-section markers; stars identify snapshots selected by fixed-start lap completion. The six configurations with the highest fixed-start lap-completion rates are highlighted.

least 95% of fixed-start agents complete a lap. Figure 3 also marks the first snapshot for which at least 10% of random-start agents pass each indicated track section.

Several settings meet the 95% criterion within five minutes, whereas others make only partial progress or remain unreliable. After selecting a configuration through this offline study, the intended workflow uses a single nominal five-minute training run for a new track.

5.2 *Stability Across Seeds*

We repeat three configurations with 32 seeds each. Figure 4 shows that 8192×128 and 10240×64 learn faster and more reliably, whereas the failure-prone 4096×256 setting reaches the 95% criterion in only about 10% of seeds. The stronger results therefore do not depend on an isolated favorable seed.

5.3 *Effect of Action Regularization*

We compare weak and strong action regularization to assess whether the deployment-oriented objective suppresses bang-bang commands at the body-rate/thrust interface. Figure 6 shows the normalized policy outputs over the first four seconds of deterministic fixed-start rollouts. Both policies use the same model, seed, rollout configuration, and optimizer settings; only the action-delta, action-magnitude, and entropy coefficients differ.

The weak setting uses $\lambda_{\Delta a} = 0.001$, $\lambda_a = 0$, and entropy coefficient 10^{-3}; the strong setting uses $\lambda_{\Delta a} = 0.25$, $\lambda_a = 0.15$, and entropy coefficient 2.5×10^{-5}. The weakly regularized policy repeatedly switches between large commands, whereas the strongly regularized policy avoids rapid switching and persistent saturation. Over the four-second interval, the mean $\|a_t - a_{t-1}\|_2^2$ decreases from 4.237 to 0.00897, and the fraction of commands near saturation, defined by $|a_{t,j}| > 0.95$, decreases from 50.00% to 1.06%. This comparison evaluates the complete regularization setting rather than the contribution of each coefficient in isolation.

5.4 *Real-World Racing Performance*

We deploy deterministic mean policies using the same state construction and body-rate/thrust interface as in training. The real system uses Weitflug, a research-oriented fork of Betaflight for agile autonomous flight [18]. The simulated trajectory in Figure 5 uses the noise-free simulator state; the real trajectory is reconstructed from the onboard EKF estimate recorded during competition. The policy and simulator run at 100 Hz. The EKF propagates IMU measurements at 1 kHz and uses camera-based gate observations for correction.

http://www.imavs.org/

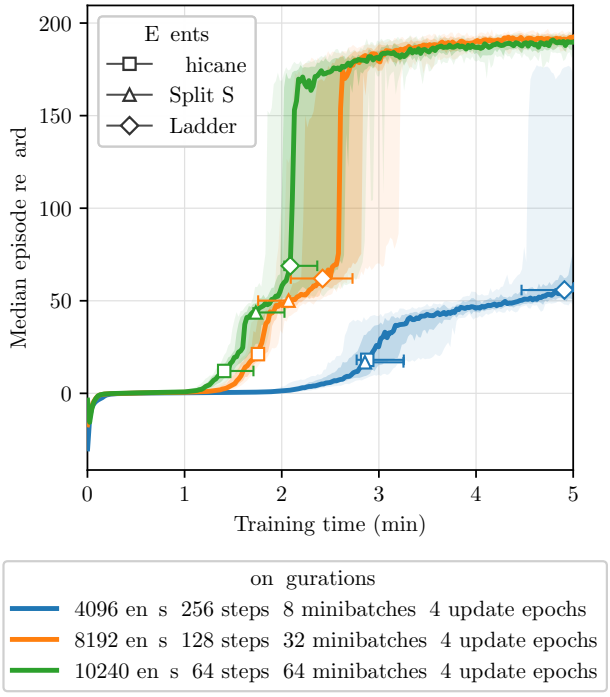


Figure 4: Seed-replication study for three representative PPO configurations. Curves and shaded quantile bands summarize median episode reward over 32 seeds per configuration; markers indicate benchmark track-section milestones.

The two policies with the highest fixed-start success rates were deployed in the Silver Cup finals. Each completed an uninterrupted two-lap run, one during a multi-drone race and the other during a timed race, with lap times agreeing within 0.1 s. The controller finished eight of ten race entries and won the Multi-Drone Silver Race [19]. The competition's VTX timing system recorded a best lap of 7.799 s, and the onboard EKF estimated a peak speed of approximately 20 m/s. The best human and autonomous laps on the same course were 6.283 s and 5.925 s, respectively.

As shown in Figure 5, the nominal and real racing lines agree well, with the largest deviation near the chicane. During this tight turn, the drone briefly points away from the next gate, reducing gate visibility and coinciding with a correction in the estimated trajectory. This behavior supports the distance-weighted gate-facing reward: progress alone is insufficient when the estimator loses informative visual references at critical track sections.

5.5 *Model Adequacy and Transfer*

This analysis evaluates robustness around the target vehicle and A2RL layout. The policy is track-specific, so a new layout requires retraining; a new vehicle also requires model identification. The five-minute workflow keeps this adaptation fast.

The largest sim-to-real mismatch occurs at the chicane in Figure 5. High speed, rapid direction changes, and reduced gate visibility make this section especially sensitive. In Figure 7, the randomized trajectories spread most strongly around the same sections, indicating that the compact simulator reveals transfer-sensitive regions despite omitting actuator-level dynamics.

The Catmull-Rom spline defines the reset distribution rather than a reference trajectory. The learned trajectories depart from it, especially around the ladder, showing that the spline broadens state coverage without imposing path following.

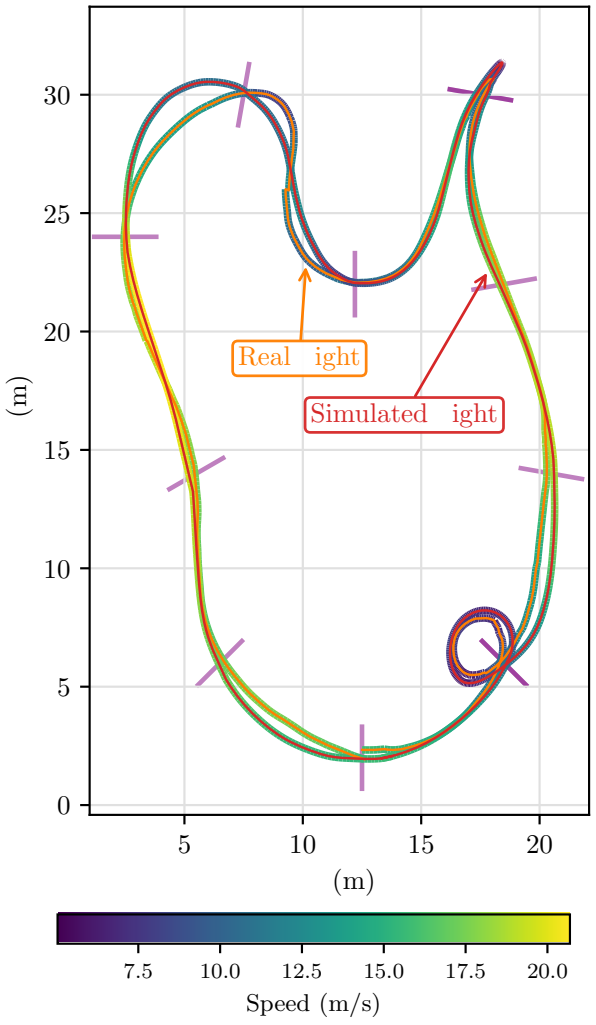


Figure 5: Top-down trajectory comparison for the same policy in nominal simulation and real flight. Color denotes speed; the real trajectory is reconstructed from the onboard EKF estimate.

http://www.imavs.org/

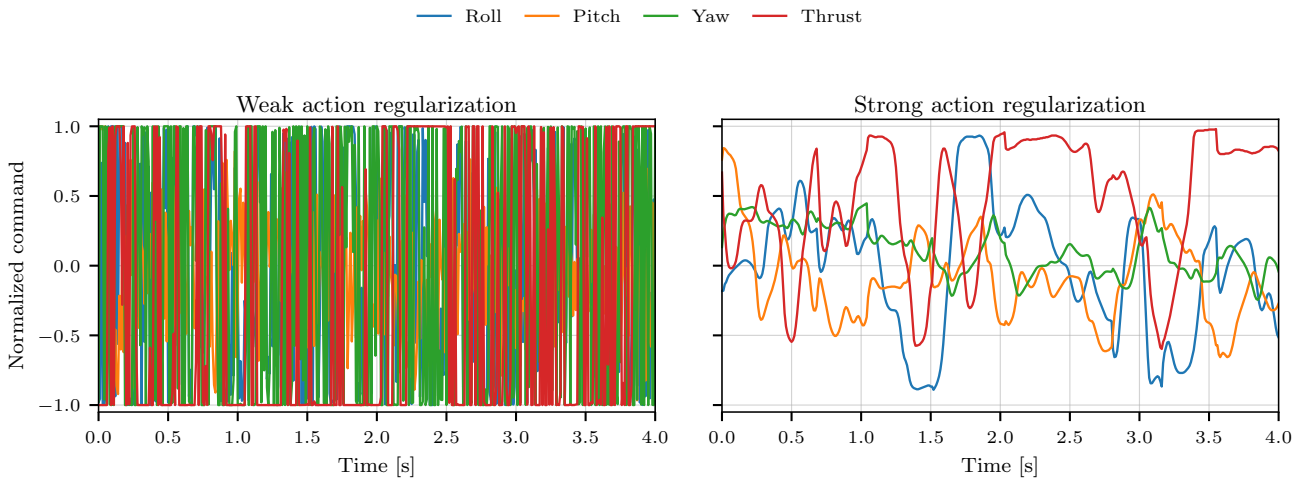


Figure 6: Normalized policy outputs over the first four seconds with weak (left) and strong (right) action regularization. Both policies reach a similar track position shortly after the split-S; stronger regularization reduces rapid switching and saturation.

http://www.imavs.org/

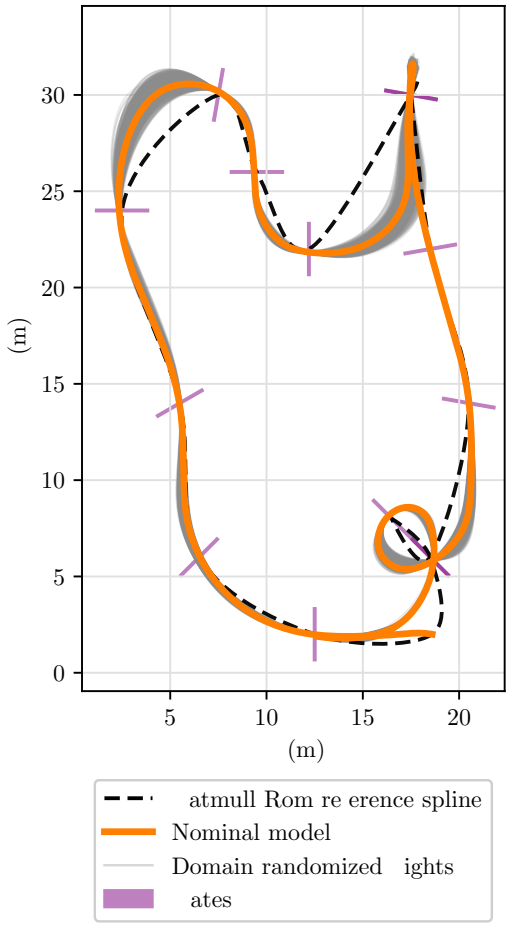


Figure 7: Nominal and domain-randomized fixed-start trajectories, with the reset spline and gate locations.

6 Conclusion

We presented a rapid sim-to-real workflow for autonomous drone racing using a compact body-rate/thrust model identified from short flight logs and massively parallel PPO in JAX. The fastest configuration reaches 32 million environment steps per second on an RTX 5070 Ti, but the 96-setting study and 32-seed analysis show that throughput alone does not determine learning quality under a five-minute budget. A matched comparison visibly suppresses rapid switching and reduces the fraction of commands near saturation, defined by $|a_{t,j}| > 0.95$, from 50.00% to 1.06%. Two selected policies transferred to the real drone, completed uninterrupted two-lap runs, and reached speeds near $20\,\mathrm{m/s}$.

The policy is optimized for a specified track: a new layout requires retraining, and a new platform also requires model re-identification. The current experiments therefore validate rapid adaptation for one vehicle and track, not cross-track or cross-platform generalization. The action comparison also changes the action-delta, action-magnitude, and entropy coefficients together. Future work will separate these effects, study the trade-off between model fidelity and training speed, and repeat the complete workflow across additional tracks and platforms.

References

[1] Hyungpil Moon, Jose Martinez-Carranza, Titus Cieslewski, Matthias Faessler, Davide Falanga, Alessandro Simovic, Davide Scaramuzza, Shuo Li, Michael Ozo, Christophe De Wagter, Guido de Croon, Sunyou Hwang, Sunggoo Jung, Hyunchul Shim, Haeryang Kim, Minhyuk Park, Tsz-Chiu Au, and Si Jung Kim. Challenges and implemented technologies used in autonomous drone racing. *Intelligent Service Robotics*, 12(2):137–148, 2019.

[2] L. Oyuki Rojas-Perez and J. Martinez-Carranza. On-board processing for autonomous drone racing: An overview. *In-*

tegration, 80:46–59, 2021.

[3] Philipp Foehn, Dario Brescianini, Elia Kaufmann, Titus Cieslewski, Mathias Gehrig, Manasi Muglikar, and Davide Scaramuzza. AlphaPilot: Autonomous drone racing. *Autonomous Robots*, 46(1):307–320, 2022.

[4] Drew Hanover, Antonio Loquercio, Leonard Bauersfeld, Angel Romero, Robert Pěnička, Yunlong Song, Giovanni Cioffi, Elia Kaufmann, and Davide Scaramuzza. Autonomous drone racing: A survey. *IEEE Transactions on Robotics*, 40:3044–3067, 2024.

[5] Stavrow A. Bahnam, Robin Ferede, Till M. Blaha, Anton E. Lang, Erin Lucassen, Quentin Missinne, Aderik E. C. Verraest, Christophe De Wagter, and Guido C. H. E. de Croon. MonoRace: Winning champion-level drone racing with robust monocular AI. *arXiv preprint arXiv:2601.15222*, 2026.

[6] Aderik Verraest, Stavrow Bahnam, Robin Ferede, Guido de Croon, and Christophe De Wagter. SkyDreamer: Interpretable end-to-end vision-based drone racing with model-based reinforcement learning. *arXiv preprint arXiv:2510.14783*, 2025.

[7] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023.

[8] Robin Ferede, Till Blaha, Erin Lucassen, Christophe De Wagter, and Guido C. H. E. De Croon. One net to rule them all: Domain randomization in quadcopter racing across different platforms. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6357–6363, 2025.

[9] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[10] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs, 2018.

[11] Angel Romero, Sihao Sun, Philipp Foehn, and Davide Scaramuzza. Model predictive contouring control for time-optimal quadrotor flight. *IEEE Transactions on Robotics*, 38(6):3340–3356, 2022.

[12] Michal Minařík, Robert Pěnička, Vojtěch Vonášek, and Martin Saska. Model predictive path integral control for agile unmanned aerial vehicles. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13144–13151, 2024.

[13] Jonas Eschmann, Dario Albani, and Giuseppe Loianno. Learning to fly in seconds. *IEEE Robotics and Automation Letters*, 9(7):6336–6343, 2024.

[14] Jonas Eschmann, Dario Albani, and Giuseppe Loianno. Data-driven system identification of quadrotors subject to motor delays. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8095–8102, 2024.

[15] Narendran Muraleedharan and Daniel S. Cohen. Modelling and simulation of UAV systems. In *Imaging and Sensing for Unmanned Aircraft Systems. Volume 1: Control and Performance*, pages 101–121. Institution of Engineering and Technology, 2020.

[16] Tim Seyde, Igor Gilitschenski, Wilko Schwarting, Bartolomeo Stellato, Martin Riedmiller, Markus Wulfmeier, and Daniela Rus. Is bang-bang control all you need? solving continuous control with bernoulli policies. In *Advances in Neural Information Processing Systems*, volume 34, pages 27209–27221, 2021.

[17] Siddharth Mysore, Bassel Mabsout, Renato Mancuso, and Kate Saenko. Regularizing action policies for smooth control with reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1810–1816, 2021.

[18] Joel Klimont, Jakob Buchsteiner, Alexander Lampalzer, and Radu Grosu. Weitflug: A research-oriented fork of Betaflight for agile autonomous flight. In *Proceedings of the 17th International Micro Air Vehicle Conference and Competition*, Strasbourg, France, 2026. To appear.

[19] Abu Dhabi Autonomous Racing League. A2RL drone championship sets the pace for AI in autonomous flight, January 2026. Accessed July 24, 2026.

http://www.imavs.org/

Precision Home Landing for Bee-Nav Robot Navigation

Dequan Ou*, Quentin Missinne, Christophe De Wagter, and Guido C.H.E. de Croon
Micro Air Vehicle Laboratory, Control and Operations, Delft University of Technology, The Netherlands

http://www.imavs.org/

Abstract

Small flying robots are an attractive platform for tasks such as greenhouse monitoring, warehouse inventory, and last-mile delivery. However, for any of these applications to be sustainable, the robot must be able to return autonomously to a recharging pad without relying on GPS or external infrastructure. The recently introduced Bee-Nav navigation strategy, inspired by honeybee learning flights, brings a small quadrotor back to within 0.5 m of its takeoff point after hundreds of metres of outbound flight, but it lacks the centimetre-level precision a charging pad requires. Here we extend Bee-Nav with a SIFT- and memory-based visual-servoing approach: a single downward SIFT view of the landing zone, captured during the same takeoff hover, is stored on board and later drives a closed-loop image-based visual-servoing descent once Bee-Nav returns the drone to the home area. Across multiple flights, the custom quadrotor lands with a mean position error of 16 ± 10 cm—a $\sim$60 % reduction in mean error and a $\sim$44 % reduction in spread relative to a camera-free output-flip detection baseline. Storing a single reference image is therefore enough to turn coarse visual homing into centimetre-precise landing, closing the loop on fully autonomous, infrastructure-free return-and-recharge cycles.

1 Introduction

Small flying robots are an increasingly attractive platform for applications such as warehouse inventory tracking and last-mile delivery [2]. For any such mission to be sustainable and easy to deploy, the robot must be able to return to a docking station for autonomous recharging without relying on external positioning infrastructure such as GPS or fiducial markers [3, 4, 5, 6]. Yet the very property that makes these robots attractive—their low payload and limited on-board compute—rules out the high-precision, map-based navigation methods that larger drones use. Micro air vehicles (MAVs) carry severely limited payloads, leaving no room for LiDAR or GPU-capable companion computers and forcing the traditional SLAM pipelines that scale with map size [7] off the table.

*Email address: d.ou@tudelft.nl

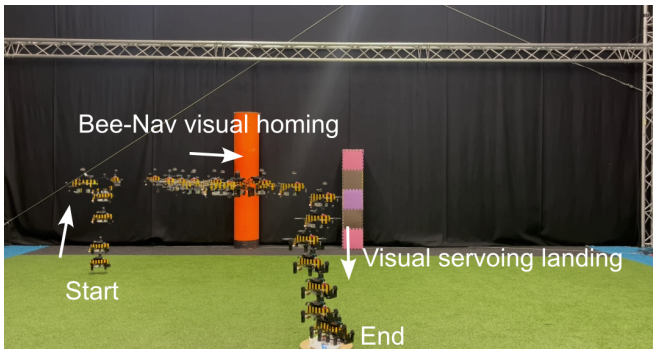


Figure 1: Time-lapse of a complete autonomous homing-and-landing flight in the CyberZoo. The drone takes off (*Start*) at a random location with random heading inside the *learned homing area* (LHA), executes Bee-Nav visual homing [1] to return to the vicinity of its takeoff point, and then performs the proposed visual-servoing-based precision landing onto a small pad (*End*).

Honeybees show that this problem is solvable with impressively minimal resources: they navigate kilometres from their hive on neural hardware of approximately one million neurons [8]. They achieve this by combining two complementary mechanisms: *path integration*, which estimates the position of the bee with respect to the hive by accumulating directions and distances travelled, and *view memory*, which maps remembered visual snapshots of the surroundings to nearby places of interest [9, 10]. In long-distance flights the two interplay: path integration provides the navigation signal while the bee is far from the nest, and view memory takes over once it is close enough to recognise familiar landmarks. The view memory itself is built during one or more short *learning flights* immediately after takeoff: while the path-integration system is still accurate, the bee deliberately samples views around the nest, and the path-integration estimate for each sample serves as the label that associates the view with its position relative to home [8, 1].

Translating this two-stage strategy onto a small drone, Bee-Nav [1] demonstrated that a 3.4-42 kB neural network trained from a single learning flight is enough to navigate back to within 0.5 m of the takeoff point from up to 110 m away in 100 % of test flights, and from up to 600 m away in the majority of outdoor flights, all using only an omnidirectional camera, an optical-flow sensor and a Raspberry Pi 4. However, 0.5 m is not enough to actually land on a recharging pad: charging contacts are typically a few centimetres

http://www.imavs.org/

wide, and the surrounding clutter (cables, neighbouring pads, infrastructure) leaves no margin for a half-metre lateral error.

In robotics, this last-half-metre gap is bridged by what is usually called *precision landing based on prior knowledge*: engineered targets such as ArUco markers or H-shaped helipads with conventional pose estimation [3, 4], classical handcrafted feature matching against a reference image, or more recently learned descriptors. We discuss these approaches in Sec. 2.3.

What is novel here is the integration of precision landing into Bee-Nav itself: by sharing its hardware and reusing its learning flight as the deployment-time reference-capture step, the combined system becomes fully autonomous from take-off to touchdown—something existing precision-landing methods, which assume a separately instrumented landing zone, do not provide on their own.

We propose a precision-landing system that extends Bee-Nav to a fully autonomous takeoff and landing pipeline. The system captures a single SIFT view of the landing zone during Bee-Nav's takeoff hover (*i.e.* during the learning flight), and then, once Bee-Nav's visual-homing phase—which runs inside the *learned homing area* (LHA), a circle of radius roughly 5–10 m centred on home [1]—has brought the drone close to its takeoff point, the landing pipeline triggers a closed-loop visual-servoing descent that drives the drone onto the centroid of the SIFT cluster seen at takeoff. The whole pipeline uses one extra downward-facing webcam and runs on the same Raspberry Pi 4 that already runs Bee-Nav.

The contributions of this work are:

1. A complete autonomous pipeline for a central-place-foraging MAV that travels out to a remote location and returns to its takeoff point with a precision sufficient for autonomous recharging, by augmenting Bee-Nav [1] with a visual-servoing precision-landing phase.

2. A SIFT-based landing-zone representation that is robust to post-takeoff hover drift, running in real time on a Raspberry Pi 4 alongside the navigation pipeline.

3. A significant reduction in landing error over a camera-free baseline that detects home arrival from overshoot-induced flips in Bee-Nav's home-direction output (*output-flip detection*), achieving centimetre-level landing accuracy without GPS, fiducial markers, or external infrastructure.

The rest of the paper is organised as follows: Sec. 2 reviews related work; Sec. 3 details the hardware, the Bee-Nav navigation strategy, and the capture-detect-servo landing pipeline; and Sec. 4 evaluates landing precision against the camera-free baseline, before Sec. 5 concludes the findings and future work.

2 RELATED WORK

2.1 Bee-Nav insect-inspired navigation

Honeybees navigate by interplaying two mechanisms: path integration drives the trajectory when the bee is far from the nest, and view memory takes over once it is close enough to recognise familiar landmarks [9, 10]. The view memory is itself built during dedicated *learning flights* immediately after takeoff, in which the path-integration estimate labels each visual snapshot with its position relative to home [8, 1]. Bee-Nav [1] ports this strategy onto a small quadrotor by training a compact convolutional homing network from learning-flight data, achieving a $\sim$0.5 m residual error to the takeoff point on a Raspberry Pi 4. We refer the reader to [1] for the full method and to Sec. 3.2 for the four-phase summary needed to follow the rest of this paper.

A half-metre residual is, however, not enough to actually land on a recharging pad. Our work closes this gap: we keep Bee-Nav's hardware and its entire navigation pipeline, and add a precision-landing phase on top.

2.2 Autonomous drone landing with prior knowledge

Most precision-landing systems instrument the pad with a fiducial pattern—H-shaped helipads [4, 3], AprilTag [5] or ArUco [6]—and reach centimetre-level pose estimation in real time on embedded hardware; the cost is operational, since every deployment site must be tagged. A second line of work, which our system belongs to, instead captures one or more reference images of the pad at deployment time and matches each live frame against them using handcrafted scale-invariant descriptors such as SIFT [11]. Reported demonstrations include natural-landmark landing [12], fully onboard monocular SIFT pipelines [13], and SVO-based elevation mapping on textured natural surfaces [14]. The failure modes of this regime are scale-variance degradation at very low altitude and sensitivity to motion blur. Recent learned descriptors (SuperPoint [15], SuperGlue [16], LightGlue [17]) and end-to-end DRL landing policies [18] improve robustness at the cost of compute that is hard to afford on Raspberry-Pi-class companions. We adopt SIFT because it gives a deterministic error signal and runs comfortably onboard, and we control its failure modes by capturing the reference during Bee-Nav's existing learning flight (Sec. 3.4) and tightly coupling it to an IBVS descent (Sec. 3.7).

2.3 Precision landing with visual servoing

Visual servoing is the standard methodology for closing the control loop in the terminal phase of autonomous UAV landing, in which visual feedback is converted directly into closed-loop velocity or attitude commands [19]. The literature divides into two canonical architectures. *Position-based visual servoing* (PBVS) first reconstructs the 3D relative pose of the target and then generates Cartesian velocity commands, yielding clean, near optimal trajectories in the workspace but inheriting a strong dependence on accurate camera calibra-

http://www.imavs.org/

tion, a known target model, and a reliable depth estimate, with no guarantee that the target remains within the field of view [3, 4]. *Image-based visual servoing* (IBVS) instead defines the error directly on the image plane and maps it to control through the interaction (image-Jacobian) matrix; this makes it robust to calibration error and keeps the target in view by construction, at the price of coupling translational and rotational motion and requiring an estimate of the feature depth [20, 21].

To avoid the depth- and calibration-sensitivity of PBVS and the coupled depth-dependent dynamics of IBVS, a family of hybrid, or 2.5D, approaches decouples the rotational and translational loops by operating partly in the cartesian workspace and partly in the image plane [22, 23]. Such schemes come at a cost, requiring a continuous homography or essential-matrix decomposition at every frame, which is comparatively expensive and yields several geometrically valid pose solutions that must be disambiguated through heuristics. For Size, Weight, and Power (SWaP)-constrained platforms such as the Raspberry-Pi-based system used here, this overhead is significantly undesirable, and pure IBVS remains the most appealing option; it offers a deterministic pixel-to-command mapping and minimal perception-to-actuation latency, with no decomposition ambiguity to resolve in flight.

A remaining difficulty, independent of the servoing architecture, is the definition of a stable geometrically meaningful target to servo towards as the apparent scale of the scene changes throughout the descent. Servoing to whichever local features happen to be detected ties the control objective to arbitrary texture and drifts as features appear and disappear. A more robust strategy is to match each live frame against a reference image of the landing zone and to project a single canonical target point, fixed once on the reference, into the current frame through the estimated homography, thereby decoupling the control objective from the specific features visible at any given instant [24, 12]. We adopt exactly this strategy: a pure IBVS descent commands a body-frame velocity towards a homography-projected target point that is fixed once during the learning-flight hover. This results in a deterministic, low-latency landing controller that fits within the SWaP budget already imposed by the navigation pipeline.

3 Methods

The proposed system combines the Bee-Nav navigation strategy with a precision-landing sub-system. The precision-landing pipeline is fully driven by a single downward-facing webcam and the on-board flight controller's state estimate (vehicle position fused from optical flow and a downward LiDAR). It is divided into two primary steps: landing trigger and visual servoing.

3.1 Hardware system

The platform is a custom-built quadrotor (Fig. 2) running PX4 on a Holybro PixHawk 6c mini flight controller.

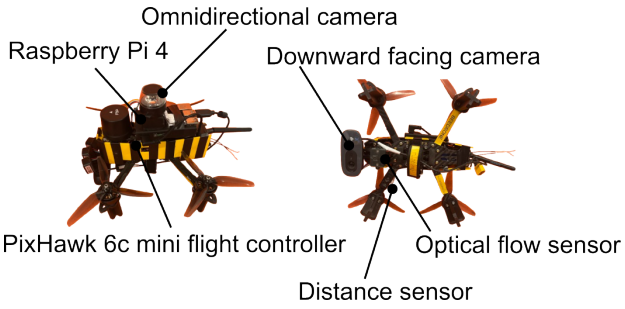


Figure 2: The custom quadrotor platform BeeDrone used in this work, shown in isometric and bottom views. The upward-pointing omnidirectional Pi-Camera v2 feeds the Bee-Nav visual-learning and homing network, while the downward-facing webcam is the sole sensor used by the precision-landing pipeline. The flight controller runs PX4; a Raspberry Pi 4 runs the vision pipeline. A downward optical-flow sensor and a distance sensor are fused by the autopilot to provide velocity and position estimates along all three axes.

The on-board companion computer is a Raspberry Pi 4 connected to the flight controller through a serial link using the micro-XRCE-DDS bridge. The Pi runs the vision pipeline on top of Raspberry Pi OS using OpenCV and a lightweight Flask HTTP layer that decouples the cadence of the vision algorithms from the ROS 2 state machine. The drone is equipped with:

- An upward-pointing fish-eye Pi-Camera v2 that feeds the Bee-Nav visual-homing network, which maps a single panoramic frame to a body-frame home vector.
- A downward-facing Logitech C270 USB webcam, the only sensor used for the precision-landing pipeline described below.
- An optical-flow sensor (PMW3901) and a downward LiDAR rangefinder (TFmini-S), fused by PX4's EKF2 to provide velocity and position estimates along all three axes.

3.2 Bee-Nav visual learning and homing

Our system extends Bee-Nav [1] with two additional phases (home capture and precision landing) that share its hardware. Bee-Nav itself consists of four phases that we refer to in the rest of this section; full details are in [1]. During a *learning flight*, the drone records pairs of (omnidirectional image, path-integration information) while flying a structured pattern within 5–10 m of the takeoff point, and a small CNN is trained to map each omnidirectional image to its associated home vector; the region within which the network can be used to reach home is called the *learned homing area* (LHA). Afterwards, when performing a task, the drone flies an *outbound* task trajectory while path integration tracks the home

direction. Once the task is completed, it follows the integrated home vector during an *inbound* flight (not ending at true home because of drift), and then enters a *visual-homing* phase inside the LHA in which the CNN is run at each step until the drone reaches home. The precision-landing pipeline inserts immediately after this visual-homing phase. In the original Bee-Nav paper, the operator triggered landing manually once the drone came within 0.5 m of the home pad; here, as a camera-free baseline, we additionally exploit a behavioural property of the same network's outputs at convergence (Sec. 3.3).

3.3 Output-flip detection (camera-free baseline)

Bee-Nav's homing network points the drone reliably toward its takeoff point throughout the return: far from home, the predicted home vector is smooth and converges, step by step, toward the takeoff point. Inside the last ~ 0.5 m, however, the drone's residual velocity carries it past home before the controller can fully decelerate, and it overshoots. Once on the other side of home, the home direction now points back the way the drone just came, so the drone reverses; on the next step it overshoots again, and so on. Repeated overshoots make the home-vector output *flip* direction every few frames as the drone dances back and forth across home—the expected behaviour of any well-tuned homing controller at convergence, not a failure of the network.

This back-and-forth pattern can itself be used as a landing trigger that requires no downward camera, no SIFT, and no extra hardware on top of Bee-Nav. We track the most recent five home-direction outputs and call a successive pair a *flip* when their angles differ by more than $90°$. When at least three of the last four pairs flip, the trigger fires: the drone enters the open-loop velocity descent of Sec. 3.6 directly and lands *in place*, accepting whatever residual error remains in the Bee-Nav inbound trajectory. We refer to this as the *output-flip detection* baseline, against which both SIFT-only and SIFT+IBVS variants are compared in Sec. 4.

3.4 Home capture at takeoff

We capture the SIFT view of the landing pad during the first hover of Bee-Nav's learning flight: at this moment the drone is hovering at takeoff altitude over (or close to) its takeoff point, and the downward webcam has a clean, sharp top-down view of whatever surface the drone will later be asked to land on. The onboard vision daemon extracts SIFT keypoints and descriptors from this frame and caches the raw image, the features, and a single 2-D target point $\mathbf{u}^\star = (u^\star, v^\star)$ in image coordinates that marks the location in the reference frame to which the drone will later be servoed.

The choice of $\mathbf{u}^\star$ is non-trivial. A naive choice would be the image centre; however, because the drone is typically not perfectly above the landing pad when the reference is captured (lateral drift during takeoff), the image centre frequently falls onto the edge or the surface next to the pad. Landing on this point therefore reproduces the takeoff drift instead of the actual pad location. Instead, we set $\mathbf{u}^\star$ to the centroid of the densest cluster of SIFT keypoints, which on a textured pad reliably falls on the pad surface itself (the pad has more high-response SIFT features than the background). The cluster centroid is computed by iterative response-weighted mean-shift:

$$\mathbf{c}_{k+1} = \frac{\sum_{i \in \mathcal{N}(\mathbf{c}_k)} r_i \, \mathbf{p}_i}{\sum_{i \in \mathcal{N}(\mathbf{c}_k)} r_i}, \tag{1}$$

where $\mathbf{p}_i$ and r_i are the position and response of keypoint i, $\mathcal{N}(\mathbf{c}_k)$ is the set of keypoints within bandwidth ρ of the current centroid, and ρ is set to 20% of the shorter image dimension. The iteration is initialised at the response-weighted mean of all keypoints and terminates either when the centroid moves less than 0.5 px between steps or after a 15-iteration cap.

3.5 SIFT-based pad detection during return

During the visual homing phase, the same vision daemon loads the cached SIFT reference at startup and runs a background detector at 5 Hz against the latest downward webcam frame. Each cycle runs SIFT detect-and-compute on the live frame, matches the live descriptors to the cached reference descriptors with a FLANN $k = 2$ search, keeps matches whose nearest/second-nearest distance ratio is below 0.75 (Lowe ratio), and fits a homography H by RANSAC. A detection is accepted only if at least 10 good matches and 10 RANSAC inliers survive with an inlier ratio above 0.4, and if the target $\hat{\mathbf{u}} = H\mathbf{u}^\star$ warped through H falls within the inner 80% of the live frame. The accepted target pixel is back-projected through the pinhole model and the current above-ground altitude h to a body-frame error $\Delta x_b = -(\hat{v} - c_y)h/f_y$, $\Delta y_b = +(\hat{u} - c_x)h/f_x$, and converted to polar (α, d). The pinhole intrinsics are derived from the C270 specification (60° diagonal FOV at 640×480) without per-unit calibration, since a small bias on d is absorbed by the closed-loop nature of the descent (Sec. 3.7). The detector runs at the chosen 5 Hz cadence with headroom for the rest of the onboard pipeline.

Each successful match is forwarded to the landing controller, which caches the most recent body-frame error along with a timestamp. This pushed cache is consumed by the landing state machine described next.

3.6 Landing state machine

The landing sub-system is a small state machine driven entirely by external set-points sent to the autopilot, so its own internal landing modes are never invoked (Fig. 4). There are two possible entry events and two operating states. An accepted SIFT detection enters the *IBVS* state, in which a body-frame XY velocity command is computed from the latest SIFT update (Sec. 3.7) and a constant descent rate is commanded vertically. The IBVS state holds until either the autopilot reports ground contact, in which case the drone disarms, or the SIFT inlier count drops below a minimum, in

http://www.imavs.org/

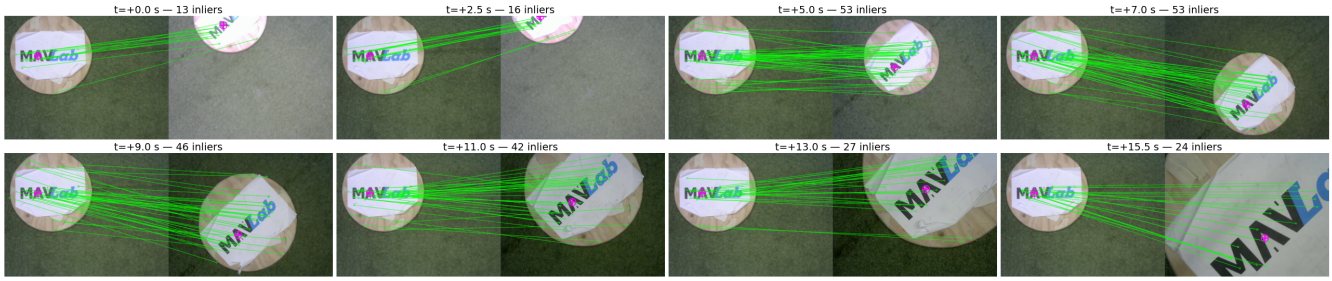


Figure 3: Onboard SIFT feature matching during the visual-servoing descent of a representative SIFT+IBVS flight, with eight frames sampled at ∼2 s intervals across the IBVS state. Each panel shows the takeoff-time landing reference (*left half*) and the live downward-camera frame (*right half*), with green lines connecting RANSAC-inlier keypoint correspondences and a magenta cross marking the pad-cluster centroid (the landing target) in each image.

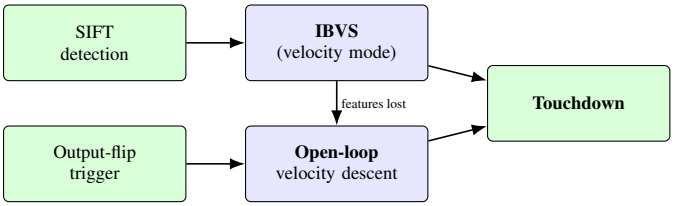


Figure 4: Landing state machine. Two entry events: an accepted SIFT detection (top-left) goes straight into velocity-mode IBVS; the camera-free output-flip detection trigger (bottom-left) goes straight into an open-loop velocity descent. The IBVS state falls through to the open-loop descent if SIFT features are lost. Both descent states terminate when PX4 reports touchdown.

which case the controller falls through to an *open-loop velocity descent*: planar position is held at the current (x, y) and the same constant downward velocity is commanded until touchdown. The output-flip detection trigger (Sec. 3.3) enters the same open-loop velocity descent directly without ever passing through the IBVS state.

3.7 *Image-based visual servoing*

The IBVS state implements a proportional image-based visual servo at the velocity set-point. Given the most recent SIFT push (Sec. 3.5)—body-frame range d and bearing α, equivalently the back-projected error $(\Delta x_b, \Delta y_b) = (d\cos\alpha, d\sin\alpha)$—the controller commands a planar speed proportional to d, rotated into the world frame by the current yaw ψ, and a constant descent rate:

$$\begin{aligned} v_{\text{des}} &= \text{clip}\big(K_v\, d,\ -v_{\max},\ +v_{\max}\big), \\ v_x^w &= v_{\text{des}}\ \cos(\psi+\alpha), \\ v_y^w &= v_{\text{des}}\ \sin(\psi+\alpha), \\ v_z &= v_d, \end{aligned} \tag{2}$$

which is published as a velocity set-point to the autopilot, whose own attitude and thrust loops track it. Entry into the IBVS state is gated by two conditions: (i) the IBVS mode must be enabled in the controller configuration; and (ii) the latest SIFT detection must be fresher than $\tau_{\text{lost}} = 300$ ms. Failure of either gate falls through to the open-loop velocity descent instead.

4 RESULTS

We evaluate three landing conditions back-to-back in the same indoor arena (CyberZoo motion-capture cage, 10×10 m): (i) the camera-free *output-flip detection* baseline of Sec. 3.3, which lands the drone in place as soon as Bee-Nav's home-direction output starts flipping from overshoot at convergence; (ii) *SIFT only*, in which a SIFT detection triggers a landing but IBVS is disabled, so the controller falls through to an open-loop velocity descent with no closed-loop XY correction; and (iii) *SIFT + IBVS*, the full system. Our focus is what SIFT+IBVS adds over Bee-Nav's own outputs; benchmarks against fiducial or learned-descriptor methods are left to future work. Eight flights per condition correspond to eight equally-spaced starting positions and directions covering the full 360° inside the LHA. All flights used the same pad and lighting; texture and illumination robustness are noted as limitations. Ground-truth trajectories were recorded from an OptiTrack motion-capture system, and only the mocap stream was used for analysis.

The final landing position relative to the target is shown in Fig. 5; the target is defined as the touchdown of a single reference SIFT+IBVS flight (whose SIFT view was acquired during the same learning flight) so that the metric is anchored to the actual visual target rather than to an arbitrary mocap-frame origin. The camera-free output-flip detection baseline lands all 8 trials but with a mean error of 0.40 ± 0.18 m and a worst case of 0.69 m—useful as a fallback when no downward camera is available, but well outside the radius a typical charging pad can tolerate. Adding the SIFT-based trigger reduces this to 0.25 ± 0.11 m, and adding velocity-mode IBVS brings it further down to 0.16 ± 0.10 m. The 0.42 m worst case of the SIFT-only condition (visible in the lower-

http://www.imavs.org/

Table 1: Per-flight landing metrics, mean ± one standard deviation across $n = 8$ flights per condition. Bold row highlights the headline precision metric. †For output-flip detection the time-to-land row is computed over the 7 of 8 trials in which the drone actually entered the 0.5 m radius before touchdown; the eighth landed at 0.60 m without ever crossing the radius.

Metric	Output-flip	SIFT only	SIFT+IBVS
Landing error [m]	$\mathbf{0.40 \pm 0.18}$	$\mathbf{0.25 \pm 0.11}$	$\mathbf{0.16 \pm 0.10}$
Time to land from 0.5 m [s]	$14.8 \pm 5.3^{\dagger}$	14.4 ± 5.9	21.9 ± 17.4
Descent path length [m]	0.85 ± 0.01	0.89 ± 0.02	0.98 ± 0.10
Descent tortuosity [-]	0.002 ± 0.001	0.037 ± 0.017	0.126 ± 0.088
Mean horizontal speed [m/s]	0.051 ± 0.027	0.037 ± 0.008	0.066 ± 0.022

left of its panel) is absent from the IBVS condition, whose worst landing is 0.30 m. An unpaired Welch's t-test confirms that SIFT+IBVS significantly reduces landing error over the output-flip baseline ($p = 0.008$, Cohen's $d = 1.62$); SIFT-only shows a similarly large effect vs. output-flip ($d = 1.01$, $p = 0.07$), and the further SIFT+IBVS-over-SIFT-only improvement is a large trend not yet significant at $n = 8$ ($d = 0.85$, $p = 0.11$).

Fig. 6 shows the visual-homing and landing trajectories. Outside the 0.5 m radius (light gray), the eight launch directions all converge on the target; inside the radius (per-condition colour), landing behaviour diverges. Output-flip detection lands wherever the trigger fires, often outside the dashed circle (one trial landed at 0.60 m without ever entering). The SIFT-driven conditions reliably enter the radius and bias the touchdown toward the target, with SIFT+IBVS producing the densest cluster.

As a complementary measure to landing-position accuracy, we report the *time-to-land from* 0.5 *m*—the elapsed time from the first sample at which the drone enters the 0.5 m radius until touchdown. This is an end-to-end view of how long each method spends in the landing zone before committing to land, and is summarised together with the landing error in Table 1.

Time-to-land is comparable between output-flip detection and SIFT-only (∼14 s on average): both commit to a descent as soon as their respective trigger fires, with no further requirement on the SIFT detection thereafter. SIFT+IBVS, in contrast, averages ∼ 22 s with much larger run-to-run variance: the median is similar at ∼ 14 s, but two of the eight trials took 45 and 54 s as the IBVS loop spent additional time waiting for the SIFT match to stabilise. The mechanism is the entry condition into the IBVS state (Sec. 3.7); failure of that condition falls through to an open-loop velocity descent, which gives up the closed-loop XY correction. To preserve precision, the system therefore effectively waits for the drone to come closer to home, where the pad spans a larger image area and matches are denser and more frequent, before and during the closed-loop descent.

5 DISCUSSION AND CONCLUSIONS

Bee-Nav alone closes a multi-hundred-metre round-trip to within 0.5 m of the takeoff point [1]; the proposed extension takes that residual to inside one charging-pad radius, making the autonomous foraging-and-recharging cycle viable on a Raspberry-Pi-class platform. The camera-free output-flip baseline lands roughly 2.5× less accurately and is best regarded as a fallback for when no downward camera is available. The closed-loop IBVS pays a real efficiency cost (a longer, more variable approach inside the landing zone) for a substantial precision gain — the expected trade-off of a closed-loop visual servo, and favourable for a fixed charging pad.

Some limitations are worth flagging:

- *Low-altitude blind zone*: below ∼ 0.3 m the pad fills the frame and the surrounding texture that anchors the homography leaves view, so residual error accumulates in the last stretch of the descent.
- *Detector cadence*: the 5 Hz SIFT loop is adequate for a static pad but would not track a moving platform or hold up under strong wind disturbance; a lighter-weight detector is needed for those regimes.
- *Detector and back-projection assumptions*: a half-visible pad, a strongly textured ground (e.g. grass), or roll and pitch during descent all shift the SIFT-cluster target off the pad; if any of these escalates to the pad leaving the camera's view entirely, or to the background out-competing the pad in keypoints, the pipeline simply cannot land precisely.
- *Scope of the evaluation*: $n = 8$ per condition gives directional coverage but limited statistical power (the SIFT+IBVS vs. SIFT-only comparison is a large-effect trend rather than a firm p-value); outdoor grass results are pending, and robustness to illumination changes, shadows, and partial pad occlusion was not evaluated.

Future work follows from these limitations. (i) A single visual system shared between Bee-Nav homing and pad detection (e.g. an omnidirectional camera with sufficient vertical FOV) would reduce the sensor count, though a panoramic reference is geometrically harder to match than a rectilinear one. (ii) Replacing SIFT with a learned local-feature pipeline (SuperPoint [15]+SuperGlue [16]/LightGlue [17]) or a lighter-weight traditional extractor would attack the motion-blur and scale-variance failure modes, and would let sub-Pi-4 platforms (STM32, Coral) — on which Bee-Nav already runs — carry the landing pipeline too. (iii) Additional low-altitude reference templates captured during takeoff (a "multi-bank" SIFT) would close the low-altitude blind zone.

In summary, we have extended Bee-Nav with a SIFT-based visual-servoing precision-landing phase that reaches centimetre-level touchdown without GPS, fiducial markers,

http://www.imavs.org/

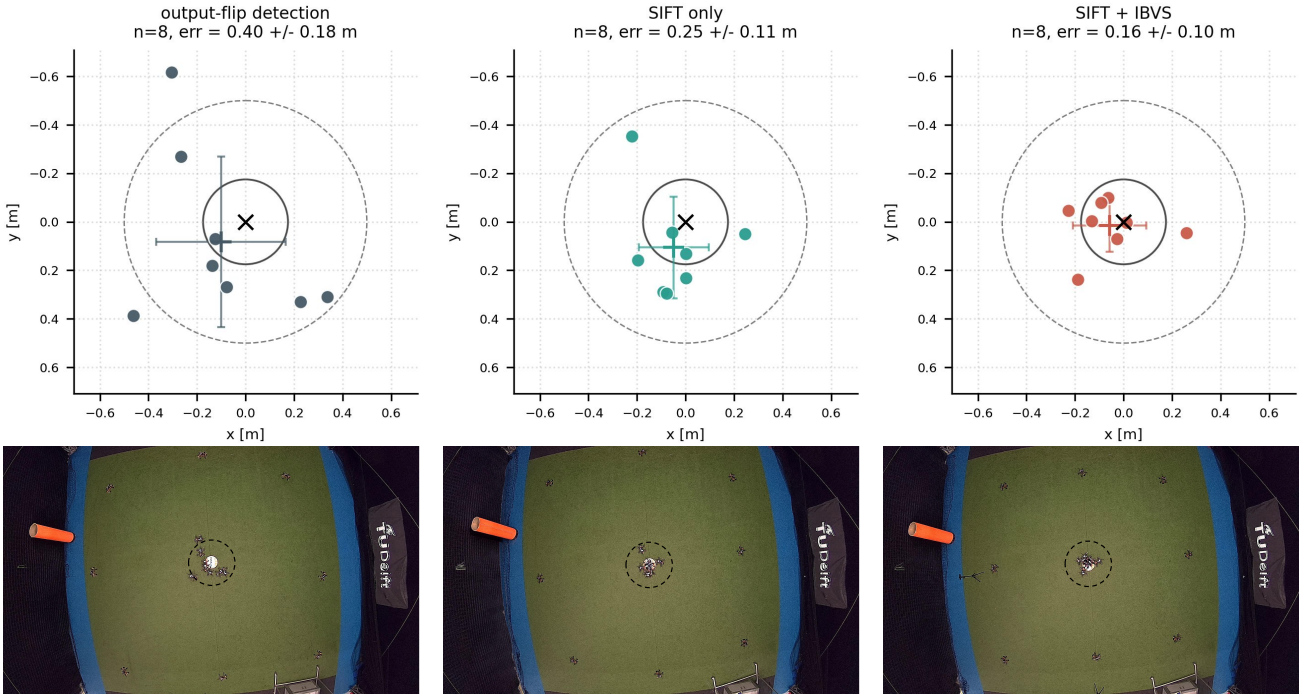


Figure 5: Touchdown precision across eight flights per condition (left: output-flip detection; middle: SIFT only; right: SIFT+IBVS). *Top row:* touchdown points expressed in the OptiTrack mocap frame relative to the visual target (black cross; defined as the touchdown of a single reference SIFT+IBVS flight). The solid inner circle (0.175 m radius) matches the size of the actual landing pad; the dashed outer circle (0.5 m radius) marks the homing-zone boundary used as the landing trigger. The coloured cross and error bars give the per-condition centroid and 1σ ellipse. Mean landing error: output-flip detection 0.40 ± 0.18 m, SIFT only 0.25 ± 0.11 m, SIFT+IBVS 0.16 ± 0.10 m. *Bottom row:* top-view time-lapse of the same eight flights inside the CyberZoo, showing the drone from launch at the edges of the cage to touchdown near the central pad; the dashed circle marks the 0.5 m homing-zone radius.

http://www.imavs.org/

or environment-side instrumentation, reusing the same hardware and learning flight Bee-Nav already performs. This makes a fully autonomous central-place-foraging round-trip practical on limited onboard compute.

Acknowledgements

This project was financed by the Dutch Research Council (NWO) under grant number 20663 of the VICI personal grant programme. We thank Taran J. Singh for his assistance with the experiments.

References

[1] Dequan Ou, Jesse J Hagenaars, Maciej R Jankowski, Michiel VM Firlefyn, Christophe De Wagter, Florian T Muijres, Jacqueline Degen, and Guido CHE de Croon. Efficient robot navigation inspired by honeybee learning flights. *Nature*, 653(8116):1039–1046, 2026.

[2] Marius Beul, David Droeschel, Matthias Nieuwenhuisen, Jan Quenzel, Sebastian Houben, and Sven Behnke. Fast autonomous flight in warehouses for inventory applications. *IEEE Robotics and Automation Letters*, 3(4):3121–3128, 2018.

[3] C. Patruno, M. Nitti, A. Petitti, E. Stella, and T. D'Orazio. A Vision-Based Approach for Unmanned Aerial Vehicle Landing. *Journal of Intelligent & Robotic Systems*, 95(2):645–664, August 2019.

[4] S. Saripalli, J.F. Montgomery, and G.S. Sukhatme. Visually guided landing of an unmanned aerial vehicle. *IEEE Transactions on Robotics and Automation*, 19(3):371–380, June 2003.

[5] Edwin Olson. AprilTag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation*, pages 3400–3407, 2011.

[6] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.

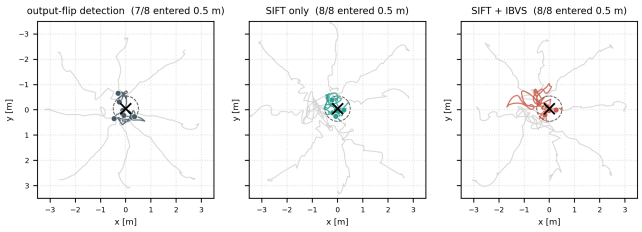


Figure 6: XY trajectories of the visual-homing and landing phases for the eight flights per condition. Outside the 0.5 m landing-zone radius around the target (dashed circle), trajectories are drawn in light gray; inside the radius, all flights are drawn in the per-condition colour, pooling the in-zone behaviour visually. Black cross: target. Filled dot at the end of each trajectory: touchdown.

[7] 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.

[8] Jacqueline Degen, Andreas Kirbach, Lutz Reiter, Konstantin Lehmann, Philipp Norton, Mona Storms, Miriam Koblofsky, Sarah Winter, Petya B. Georgieva, Hai Nguyen, Hayfe Chamkhi, Uwe Greggers, and Randolf Menzel. Exploratory behaviour of honeybees during orientation flights. *Animal Behaviour*, 102:45–57, 2015.

[9] Thomas Stone, Barbara Webb, Andrea Adden, Nicolai Ben Weddig, Anna Honkanen, Rachel Templin, William Wcislo, Luca Scimeca, Eric Warrant, and Stanley Heinze. An anatomically constrained model for path integration in the bee brain. *Current Biology*, 27(20):3069–3085, 2017.

[10] BA Cartwright and TS Collett. Landmark maps for honeybees. *Biological cybernetics*, 57(1):85–93, 1987.

[11] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[12] Andrea Cesetti, Emanuele Frontoni, Adriano Mancini, Primo Zingaretti, and Sauro Longhi. A vision-based guidance system for UAV navigation and safe landing using natural landmarks. *Journal of Intelligent & Robotic Systems*, 57(1):233–257, 2010.

[13] Shaowu Yang, Sebastian A. Scherer, and Andreas Zell. An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle. *Journal of Intelligent & Robotic Systems*, 69(1):499–515, 2013.

[14] Christian Forster, Matthias Faessler, Federico Fontana, Manuel Werlberger, and Davide Scaramuzza. Continuous on-board monocular-vision-based elevation mapping applied to autonomous landing of micro aerial vehicles. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 111–118, 2015.

[15] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperPoint: Self-supervised interest point detection and description. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 337–349, 2018.

[16] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4937–4946, 2020.

[17] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. LightGlue: Local feature matching at light speed. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 17581–17592, 2023.

[18] Alejandro Rodriguez-Ramos, Carlos Sampedro, Hriday Bavle, Paloma de la Puente, and Pascual Campoy. A deep reinforcement learning strategy for UAV autonomous landing on a moving platform. *Journal of Intelligent & Robotic Systems*, 93(1):351–366, 2019.

[19] François Chaumette and Seth Hutchinson. Visual servo control. i. basic approaches. *IEEE Robotics & Automation Magazine*, 13(4):82–90, 2006.

[20] Daewon Lee, Tyler Ryan, and H Jin Kim. Autonomous landing of a vtol uav on a moving platform using image-based visual servoing. In *2012 IEEE International Conference on Robotics and Automation*, pages 971–976. IEEE, 2012.

[21] Pedro Serra, Rita Cunha, Tarek Hamel, David Cabecinhas, and Carlos Silvestre. Landing of a quadrotor on a moving target using dynamic image-based visual servo control. *IEEE Transactions on Robotics*, 32(6):1524–1535, 2016.

[22] Ezio Malis, Francois Chaumette, and Sylvie Boudet. 2 1/2 d visual servoing. *IEEE Transactions on Robotics and Automation*, 15(2):238–250, 1999.

[23] Francois Chaumette and Seth Hutchinson. Visual servo control. ii. advanced approaches [tutorial]. *IEEE Robotics & Automation Magazine*, 14(1):109–118, 2007.

[24] Selim Benhimane and Ezio Malis. Homography-based 2d visual tracking and servoing. *The International Journal of Robotics Research*, 26(7):661–676, 2007.

http://www.imavs.org/

VLN on the Fly: An Onboard Vision-Language Navigation Stack for Aerial Robots

Marco S. Tayar[†, *], Felipe Tommaselli[*], Gianluca Capezutto[*], Pedro Antonio Rabelo Saraiva[*], Pedro H. V. de Freitas, Lucas Kido, Guilherme Sonego, Ricardo V. Godoy, and Marcelo Becker
University of São Paulo (USP), Brazil

[*]These authors contributed equally to this work. [†] Corresponding author: marcotayar@usp.br

http://www.imavs.org/

Abstract

Running vision-language navigation fully onboard an aerial robot is hard, since grounding, planning, and control must share limited compute and a single-stage error is difficult to isolate in flight. End-to-end aerial policies fuse these stages into one network, giving up the observability and safety checks a modular stack keeps available. We propose VLN on the Fly, an onboard stack that keeps grounding, planning, and control as separate, inspectable stages. A quantized VLM grounds an instruction to a coarse image cell, depth lifts it to a 3D goal, a fast B-spline planner returns a feasible trajectory, and a pretrained reinforcement learning policy tracks it to motor commands across quadrotors. Across 15 onboard flights over three everyday referents in a controlled indoor volume, the stack reaches the target in 13 of 15 trials with 5.72 cm mean goal error and 39.3% average GPU utilization. In 6 additional cluttered-environment trials, the stack tracks collision-free trajectories under onboard perception gating.

1 Introduction

Aerial robots are increasingly being deployed in human-centered indoor environments, including warehouses, offices, inspection sites, and residential spaces. Indoor deployment introduces GPS denial, cluttered geometry, narrow passages, and layouts built around people or ground vehicles rather than autonomous flight. Operation under such conditions depends on the platform's robustness and the ability of the underlying task-completion models to generalize beyond a fixed environment. Language-conditioned control is promising in this setting because natural-language instructions allow a human to specify task intent directly in a new scenario, without re-engineering the system for each task.

Vision-Language-Action (VLA) models have been used to address this need, with a single end-to-end policy trained to map raw observations and instructions directly to control commands. Applied to navigation, this class is commonly referred to as Vision-Language Navigation (VLN) [1–4], and recent aerial works such as AerialVLA [5], AutoFly [6], and Singer [7] point in promising directions. However, the compositional generalization behind the success of language models has proven harder to achieve in physical autonomy [8]. For aerial robots, each new instruction must be grounded not only in semantic context but also in feasible motion under vehicle dynamics and safety constraints. When grounding, spatial reasoning, and control are encoded inside a single end-to-end policy, limited observability makes failures harder to localize and safety constraints harder to inspect or enforce through standard planning mechanisms such as collision checks, feasibility tests, and recovery behaviors [9].

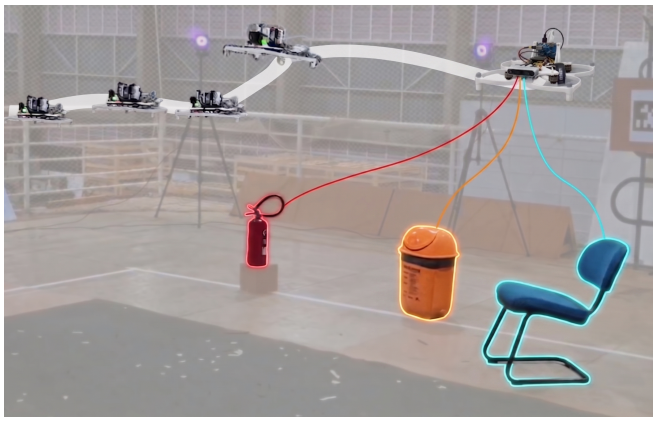


Figure 1: **VLN on the Fly**, an onboard vision-language navigation stack that grounds language instructions and plans and executes trajectories toward named referents, here a fire extinguisher, a trash bin, or a chair.

Recent literature shows that decoupling these models enables desirable behaviors in navigation stacks [10, 11], since separate modules remain observable, reusable path-planning safety guards remain available, and changes to one component of the stack do not require retraining the entire policy. For a decoupled approach, integrating the hierarchical stack, particularly with the flight controller's inner modules, remains a valid concern underexplored in the literature. Concretely, we address three concerns: (i) safety, (ii) hierarchical complexity, and (iii) onboard compute, since running a local Vision-Language Model (VLM) together with a full stack onboard is inherently challenging.

http://www.imavs.org/

In this work, we propose **VLN on the Fly**, an onboard stack for navigation on aerial vehicles (Fig. 1). We embed a quantized VLM (Qwen-3.5-2B [12]) for coarse grid grounding, where the model identifies the image region containing the target object from the language instruction and visual scene. The selected region is combined with depth information to estimate a 3D goal, which is passed to EGO-Planner [13] for B-spline trajectory generation. To overcome the low-level complexity of the hierarchical pipeline, we leverage the pretrained RAPTOR policy [14], which maps a position setpoint and the proprioceptive data directly to motor outputs. To maintain safety across this hierarchy, we validate the projected goal against the operating bounds before planning, while the planner's occupancy map supports collision-aware trajectory generation. We investigate these questions through system design, real-flight evaluation, and per-module characterization. Our contributions are:

- An onboard decoupled aerial VLN stack combining a quantized VLM for open-vocabulary grounding, EGO-Planner for 3D B-spline planning, and pretrained RAPTOR for low-level control, without end-to-end training or offboard compute.
- A lightweight safety supervisor, using a finite-state machine to reject projected goals outside the operating bounds and to gate trajectory execution based on planner feasibility from the onboard occupancy map.
- A real-flight feasibility evaluation of the full onboard stack on an open-vocabulary object-goal task over three referents, and a per-stage failure attribution enabled by the modular design.

2 Related Work

Aerial VLN has largely been end-to-end, a single network mapping instructions and egocentric observations to actions, from the AerialVLN benchmark [15] to recent open-vocabulary models [6, 7], with VLA-AN [16] carrying such a policy fully onboard. This coupling forfeits the observability and path-planning safety guards of a conventional stack and ties the policy to one platform. In contrast, VLN on the Fly keeps language grounding, metric planning, and low-level control as separate stages, enabling onboard VLN while preserving inspectable interfaces.

Decoupled VLNs split semantic reasoning from control to recover those guards. DualVLN [11] separates high and low-level execution, while See-Point-Fly [17] and Fly0 [18] ground an instruction to an image point, the latter projects it to a 3D target with depth for a geometric planner, the same mid-stack we adopt. AirHunt [19] keeps the slow VLM from bottlenecking the planner, and OnFly [20] runs the VLM fully onboard on our class of embedded computers, confirming the recipe is both effective and onboard-capable. We adopt the same decoupling principle, but replace pixel-level grounding with coarse-grid grounding and connect the VLM-to-planner interface to a learned flight policy.

Reinforcement Learning Policies map vehicle state directly to actuator commands at the high control rates required for agile flight, around 100 Hz, as shown in champion-level drone racing [21] or autonomous inspection [22]. However, many learned controllers specialize in a single airframe and require renewed system identification or retraining after hardware changes. RAPTOR [14] addresses this issue with a foundation policy that adapts across quadrotors zero-shot. We use RAPTOR as the low-level stage of the decoupled VLN stack, feeding position setpoints directly from the planner and avoiding a hand-tuned position-attitude cascade.

3 System Design

Our framework presents a modular, hierarchical, and fully onboard VLN architecture with three processing stages coordinated by a safety supervisor (Fig. 2). The grounding stage converts the instruction and current observation into a 3D goal, EGO-Planner generates a feasible trajectory, and RAPTOR tracks the resulting setpoints. Explicit goals and setpoints keep the interfaces observable, while the supervisor validates candidate goals and gates trajectory execution.

3.1 VLM-based Goal Grounding

We use Qwen-3.5-2B quantized to INT4 [12], reducing memory use enough for onboard inference. At each VLM query, the model takes the egocentric RGB frame and the instruction, then returns one of nine named regions from a 3×3 image grid together with a confidence score, or reports that the target is not visible. The grid is defined only in the text prompt and is not rendered onto the image. The returned region is mapped to the corresponding cell during post-processing. Restricting the output to nine region names removes pixel coordinates, making the result easier to check.

For the returned cell, the median valid depth d_{cell} is assigned to its center pixel $(u_{\text{cell}}, v_{\text{cell}})$. The corresponding point in the camera frame is obtained through the back-projection in (1):

$$\mathbf{p}_{\text{camera}} = d_{\text{cell}} \begin{bmatrix} (u_{\text{cell}} - c_x)/f_x \\ (v_{\text{cell}} - c_y)/f_y \\ 1 \end{bmatrix}, \tag{1}$$

where f_x and f_y are the camera focal lengths, and (c_x, c_y) is the principal point. The current pose then transforms the point into the map frame to obtain the planner goal. Using the cell median reduces sensitivity to isolated invalid or noisy depth readings. If the cell contains no valid depth within the sensor range, the goal is rejected.

The goal altitude is constrained to a narrow band around the current flight altitude, keeping the goal within a safe vertical range and avoiding climbs or descents caused by depth noise. Between VLM queries, the planner and policy continue running at their own rates.

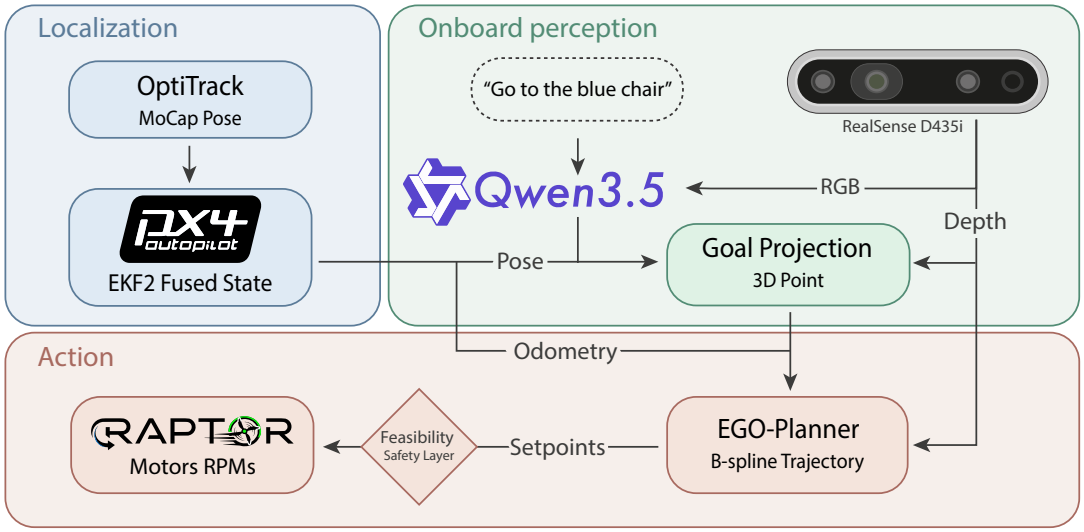


Figure 2: **Overview of the VLN on the Fly.** The RGB-D camera and language instruction are processed onboard by Qwen3.5 to project a 3D goal, EGO-Planner generates a feasible B-spline trajectory, and RAPTOR tracks the resulting setpoints as motor commands. OptiTrack pose is fused by EKF2 for localization in the experimental setup, while the safety layer gates motion.

http://www.imavs.org/

3.2 *Safety Supervisor*

The safety supervisor controls the handoff between VLM grounding and EGO-Planner. At take-off, the initial vehicle position defines the center of a bounded operating volume. A candidate goal (x_g, y_g, z_g) is accepted only when

$$|x_g - x_0| \le b_x, \qquad |y_g - y_0| \le b_y, \qquad |z_g - z_0| \le b_z, \quad (2)$$

where (x_0, y_0, z_0) denotes the initial vehicle position and b_x, b_y, and b_z define the configured limits along each axis. Goals violating (2) are rejected before planning.

Before goal publication, the supervisor requires three consecutive predictions for the same grid region, each above the configured confidence threshold. Only predictions with a valid projected goal enter the consensus process. After acceptance, one goal remains active until EGO-Planner reports completion, rejection, or failure. Nearby proposals and rapid updates are suppressed to prevent unstable commands between VLM queries.

3.3 *3D Planning*

The validated 3D goal is passed to EGO-Planner [13], which returns a smooth, dynamically feasible trajectory. The planner does not require a Euclidean Signed Distance Field (ESDF), allowing local, on-demand collision evaluation without constructing a full distance field. The lower computational cost allows EGO-Planner to run alongside the VLM on the same onboard computer. EGO-Planner represents trajectories as uniform B-splines and plans natively in 3D.

The B-spline convex-hull property allows velocity, acceleration, and collision constraints to be applied through the control points. The corresponding limits are configured for each airframe, defining how the platform's motion limits are passed to the planner. Local support also confines each control-point update to a short segment of the trajectory.

After goal validation, EGO-Planner checks the trajectory's feasibility against a local occupancy map built from the depth camera and the current pose. The map is entered into the collision cost before the optimized B-spline is sampled into position setpoints. Replanning refreshes the setpoints.

3.4 *Low-level Policy*

A conventional flight-control stack maps planner setpoints to motor commands through position, velocity, attitude, and rate controllers, each tuned to the drone's mass, inertia, motor constants, and thrust limits. RAPTOR [14] replaces the cascade with a pretrained policy that maps the setpoint-relative state directly to actuator commands.

RAPTOR receives the position setpoints from Section 3.3 relative to the current vehicle state. A lightweight tracking bridge samples the planned path as a moving reference and provides position setpoints with velocity feedforward. After replanning, the reference resumes from the path point nearest the current vehicle position, reducing discontinuities between successive trajectories. Updating the stream after replanning ensures the policy tracks the latest trajectory. RAPTOR handles low-level control through a network with 2,084 parameters and three recurrent layers, running directly on the flight controller without retraining for our airframe.

4 Experimental Results

In this section, we evaluate VLN on the Fly as an onboard aerial VLN stack for open-vocabulary object-goal navigation. We first describe the platform and experimental setup, then evaluate goal-reaching performance across 15 real flights and

http://www.imavs.org/



Figure 3: **Experimental Setup.** **(a)** Controlled indoor flight volume for the open-vocabulary object-goal trials, with the quadrotor, marked operating bounds, and everyday referents. **(b)** Cluttered setup for the path-planning trials (Sec. 4.3), with a table, chair, tool case, and a freestanding vertical panel placed between the take-off pose and the trash-bin referent.

profile onboard VLM latency. We next analyze the perception design through grid-resolution, depth-readout, and referent-discrimination studies. Finally, we test the complete stack in cluttered scenes to assess goal validation, obstacle-aware planning, and trajectory tracking.

We deploy VLN on the Fly on MIRA [23], an open-source quadrotor equipped with an Intel RealSense D435i RGB-D camera, an onboard Jetson Orin NX, and a Pixhawk 6C flight controller. An OptiTrack PrimeX 41 motion-capture system provides pose estimates used as odometry.

4.1 *Performance Analysis*

Trials are conducted as shown in Fig. 3, with three everyday referents: a trash bin, a chair, and a fire extinguisher. For each referent, we run the full stack 5 times, for a total of 15 onboard flights from the same take-off pose. Each flight receives a single instruction naming the target. The scene layout remains fixed during each run.

As reported in Table 1, our method reached the commanded referent in 13 of 15 flights (87% success rate, with a Wilson 95% confidence interval of 0.62–0.96). The trash bin succeeded in all five runs, ending directly above the referent every time (zero goal error) even though the executed paths curved on the way in. The chair and fire extinguisher each failed once, and in both cases the vehicle approached the correct object but stopped short of the 20 cm radius once the target left the camera view, preventing goal refinement.

Table 1: Open-Vocabulary Object-Goal Navigation

Referent	**Success rate ↑**	**Goal err. (cm) ↓**	**Track err. (cm) ↓**	**Avg. GPU (%)**
Trash bin	1.00	0.00	34.96	42.2
Chair	0.80	10.07	26.13	37.3
Fire extinguisher	0.80	7.08	37.64	38.4
Overall	**0.87**	**5.72**	**32.91**	**39.3**

The qualitative run in Fig. 4 exposes the main interfaces of the stack during a successful run. The VLM output is correct at the coarse-region level, supporting our choice to avoid pixel-level grounding. However, the main residual error appears downstream, as the limited RGB-D range and local inflation in the occupancy map can bias the planner away from a straight approach, as seen in the curved projected trajectory.

Beyond the GPU utilization of Table 1, we profile the grounding stage directly from the onboard inference records (Table 2, N=1865 queries). On the Jetson Orin NX the quantized 2B VLM completes a grounding query in a median of 0.79 s (95th percentile 0.88 s), which comfortably sustains the 0.5 Hz query rate and leaves the planner and policy to run uninterrupted in between. Time-to-first-token, which covers vision encoding and prompt prefill, dominates and is nearly constant at 0.60 s, the total time varies only with the short region output, so latency is stable across queries.

Table 2: Onboard VLM Inference Timing

Stage	**Median**	**95th pct.**	**Max**
Time-to-first-token	0.60 s	0.66 s	0.72 s
Total inference	0.79 s	0.88 s	1.12 s

4.2 *Perception Ablation Study*

To isolate the effect of otherwise untracked hyperparameters and to motivate our design choice, Table 3 presents an offline analysis of the perception layer. We replayed the logged onboard depth frames using the same cell-selection and projection pipeline as in flight. In addition to the valid-goal rate against the ground-truth position, we report the inter-frame goal-selection jitter.

A 1×1 grid provides stable depth estimates but no image-plane localization and therefore cannot spatially distinguish multiple referents. Among grids that preserve spatial localization, 3×3 achieves a valid-goal rate of 1.00 with substantially lower jitter than 5×5, motivating its adoption as our default configuration.

Taking the grid to its limit, a single pixel, removes the region abstraction and makes explicit why we ground at the cell level (bottom of Table 3). Back-projecting the goal from

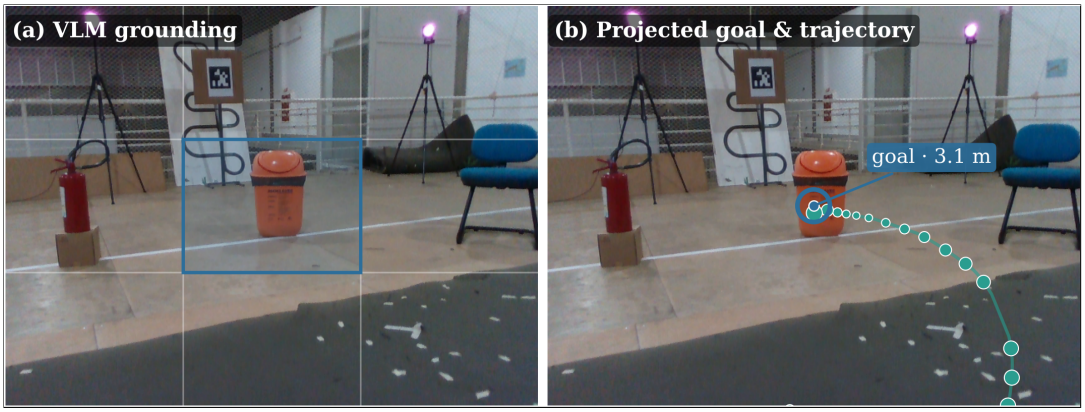


Figure 4: **Qualitative Grounding and Planning Result.** A single onboard frame from a representative successful run for the instruction *"go to the trash bin"*, with the fire extinguisher and chair present as distractors. **(a)** VLM grounding over the image grid: the selected center cell (blue) localizes the referent. **(b)** The same frame as a navigation command: the back-projected 3D goal at the selected cell's ray, and the executed onboard trajectory projected into the camera view.

Table 3: Grounding Resolution Ablation

Region grid	Valid-goal rate ↑	Selection jitter (m) ↓
1×1	0.94	0.07
3×3 (ours)	1.00	0.70
5×5	1.00	1.27
Single pixel	0.15	1.16

Table 4: VLM Region Selection by Prompted Referent

Prompted referent	Not visible	Modal region	Share
Chair (left)	53%	Top-left	86%
Trash bin (center)	44%	Center	90%
Fire extinguisher (absent)	66%	Center	79%

one image pixel yields a valid depth reading in only ∼15% of frames (invalid in 85%), since a lone pixel frequently lands on a low-texture, reflective, or out-of-range surface, and where it is valid, it can read across an object boundary and back-project from the background behind the referent, differing from the cell median by up to 0.8 m. The selected 3×3 cell median avoids these failures.

To further support the open-vocabulary claim, we replayed identical onboard frames through the VLM, varying only the referent name in the prompt (N=1865 queries, Table 4). On frames where the model commits to a region, the two present referents map to disjoint, scene-consistent locations, the chair to the top-left (86%) and the trash bin to the center (90%). The absent referent is correctly reported as not visible in 66% of frames, but when it does commit, it defaults to the center (79%), the trash bin's region. This hallucination mode does not affect the discrimination between present referents, but it suggests trusting not-visible reports over commitments to out-of-scene targets. Although the VLM exhibits a high false-negative rate for present objects (44–53% not visible), the tracking and projection pipeline successfully bridges these intermittent perception gaps.

4.3 Cluttered-Environment Evaluation

The trials above isolate grounding accuracy in a largely open volume. To probe the stack under the cluttered conditions typical of indoor deployment, we ran a small set of additional onboard flights ($n = 6$) toward the trash-bin referent with obstacles (a chair, a tool case, and mocap tripods) placed between the take-off pose and the target. Using the onboard occupancy map, EGO-Planner produced collision-free B-spline trajectories that RAPTOR tracked to the referent. In three of the six runs the stack reached the referent collision-free, ending 0.00–0.183 m from the goal. In the other two the safety layer never released a goal that satisfied its depth and bounds checks, so no trajectory was executed and no collision occurred. In the remaining run (Run 3), the vehicle successfully avoided obstacles but terminated just outside the success radius (0.21 m from the goal) due to high tracking deviation. This demonstrates that the same onboard stack extends from open-volume goal reaching to cluttered navigation without changes, and that the goal-validation interface degrades to a safe no-commit. Fig. 5 shows the planned and flown trajectories for a representative run, and Table 5 reports per-run errors measured from trajectory generation to the first goal-reach. Across the four executed runs the flown path stayed close to the planned B-spline, with a mean tracking error of 17.6 cm (per-run means 10.9–29.9 cm), altitude was held near the commanded band, so the horizontal path carried the navigation while the vehicle navigated the clutter.

5 Limitations

As with many open-vocabulary systems, the generalization of our results is not fully characterized. While experi-

http://www.imavs.org/

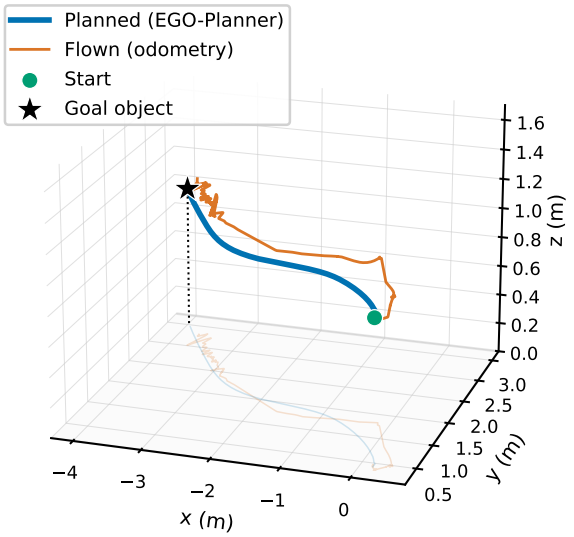


Figure 5: **Planned vs. Flown Trajectory in Clutter.** EGO-Planner's collision-free B-spline (blue) runs from take-off (green) to the goal object (star), and RAPTOR tracks it (orange). Altitude is held near the commanded band, faint floor shadows show the horizontal path. The window spans trajectory generation to the first goal-reach.

ments demonstrate the feasibility of the proposed VLN tasks, gains in modularity and safety come at the cost of integration trade-offs. In particular, the target remains fixed between goal queries, so a grounding error can persist until the next query, an asynchronous design could refresh it sooner. Grounding also operates on the current egocentric frame alone, with no spatial memory, so a referent outside the field of view must first be brought into view before it can be grounded, which bounds the contextual and sequential task families. Finally, depth back-projection proved to be limited in practice by the stereo sensor's range and by invalid depth regions.

To isolate the present contributions, we made two simplifying assumptions that future work could relax: fixed-altitude flight and external odometry. The z-clamp fixes the altitude per environment, so the planner's full 3D capability is not exercised. Moreover, all trials are conducted within a single OptiTrack volume from a known initial pose, without evaluating arbitrary initialization or relocalization.

6 Conclusion

We presented **VLN on the Fly**, an onboard decoupled aerial VLN stack that grounds natural-language instructions into 3D goals with a quantized VLM, plans collision-aware trajectories with EGO-Planner [13], and executes position setpoints through the pretrained RAPTOR policy [14]. The decoupled design preserves intermediate observability, allowing grounding, goal projection, planning, and control errors

Table 5: Goal and Tracking Errors for Cluttered Flights

Run	Goal err. (cm) ↓	Mean track. err. (cm) ↓	p90 track. err. (cm) ↓
1	0.00	14.5	18.6
2	18.3	15.1	26.4
3	20.7	29.9	79.1
4	0.00	10.9	21.2
Mean	**9.75**	**17.6**	**36.3**

to be inspected separately during real flight. Our experiments characterize open-vocabulary object-goal navigation in a controlled indoor volume, demonstrating onboard execution while highlighting remaining sensitivity to depth range, metric projection, and planner-interface effects. Complementary sensing, longer-range depth, and tighter coupling between perception uncertainty and trajectory generation could improve robustness in less constrained environments and provide a clearer path toward longer-horizon aerial VLN.

7 Acknowledgments

This work was supported in part by the São Paulo Research Foundation (FAPESP), Grants #2025/20858-7 and #2025/22381-3; in part by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Grant 308092/2020-1; and by Petróleo Brasileiro S/A - Petrobras, using resources from the ANP R&D clause, in partnership with the University of São Paulo (USP) and the Fundação de Apoio à Física e à Química (FAFQ), under Cooperation Agreements #2023/00016-6 and #2023/00013-7.

References

[1] Hongyi Zhou, Weiran Liao, Xi Huang, Yucheng Tang, Fabian Otto, Xiaogang Jia, Xinkai Jiang, Simon Hilber, Ge Li, Qian Wang, Ömer Yağmurlu, Nils Blank, Moritz Reuss, and Rudolf Lioutikov. Beast: Efficient tokenization of b-splines encoded action sequences for imitation learning. In *Advances in Neural Information Processing Systems*, volume 38, pages 172934–172959, 2025.

[2] Moritz Reuss, Hongyi Zhou, Marcel Rühle, Ömer Erdinç Yağmurlu, Fabian Otto, and Rudolf Lioutikov. Flower: Democratizing generalist robot policies with efficient vision-language-action flow policies, 2025.

[3] Dhruv Shah, Ajay Sridhar, Arjun Bhorkar, Noriaki Hirose, and Sergey Levine. GNM: A general navigation model to drive any robot. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7226–7233, 2023.

[4] Haoran Liu, Weikang Wan, Xiqian Yu, Minghan Li, Jiazhao Zhang, Bo Zhao, Zhibo Chen, Zhongyuan Wang,

http://www.imavs.org/

Zhizheng Zhang, and He Wang. Navid-4d: Unleashing spatial intelligence in egocentric RGB-D videos for vision-and-language navigation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10607–10615, 2025.

[5] Peng Xu, Zhengnan Deng, Jiayan Deng, Zonghua Gu, and Shaohua Wan. Aerialvla: A vision-language-action model for UAV navigation via minimalist end-to-end control, 2026.

[6] Xiaolou Sun, Wufei Si, Wenhui Ni, Yuntian Li, Dongming Wu, Fei Xie, Runwei Guan, He-Yang Xu, Henghui Ding, Yuan Wu, Yutao Yue, Yongming Huang, and Hui Xiong. Autofly: Vision-language-action model for UAV autonomous navigation in the wild, 2026.

[7] Maximilian Adang, JunEn Low, Ola Shorinwa, and Mac Schwager. SINGER: An onboard generalist vision-language navigation policy for drones, 2025.

[8] Yue Zhang, Ziqiao Ma, Jialu Li, Yanyuan Qiao, Zun Wang, Joyce Chai, Qi Wu, Mohit Bansal, and Parisa Kordjamshidi. Vision-and-language navigation today and tomorrow: A survey in the era of foundation models. *Transactions on Machine Learning Research*, 2024.

[9] Joonkyung Kim, Wenxi Chen, Davood Soleymanzadeh, Yi Ding, Xiangbo Gao, Zhengzhong Tu, Ruqi Zhang, Fan Fei, Sushant Veer, Yiwei Lyu, Minghui Zheng, and Yan Gu. Modular safety guardrails are necessary for foundation-model-enabled robots in the real world, 2026.

[10] Abdelrhman Werby, Chenguang Huang, Martin Büchner, Abhinav Valada, and Wolfram Burgard. Hierarchical open-vocabulary 3D scene graphs for language-grounded robot navigation. In *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*, 2024.

[11] Meng Wei, Chenyang Wan, Jiaqi Peng, Xiqian Yu, Yuqiang Yang, Delin Feng, Wenzhe Cai, Chenming Zhu, Tai Wang, Jiangmiao Pang, and Xihui Liu. Ground slow, move fast: A dual-system foundation model for generalizable vision-and-language navigation, 2025.

[12] Qwen Team. Qwen3.5: Towards native multimodal agents, 2026.

[13] Xin Zhou, Zhepei Wang, Hongkai Ye, Chao Xu, and Fei Gao. EGO-Planner: An ESDF-free gradient-based local planner for quadrotors. *IEEE Robotics and Automation Letters*, 6(2):478–485, 2021.

[14] Jonas Eschmann, Dario Albani, and Giuseppe Loianno. RAPTOR: A foundation policy for quadrotor control. *Science Robotics*, 11(114):eaec1481, 2026.

[15] Shubo Liu, Hongsheng Zhang, Yuankai Qi, Peng Wang, Yanning Zhang, and Qi Wu. Aerialvln: Vision-and-language navigation for UAVs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15384–15394, 2023.

[16] Yuze Wu, Mo Zhu, Xingxing Li, Yuheng Du, Yuxin Fan, Wenjun Li, Zhichao Han, Xin Zhou, and Fei Gao. VLA-AN: An efficient and onboard vision-language-action framework for aerial navigation in complex environments, 2025.

[17] Chih Yao Hu, Yang-Sen Lin, Yuna Lee, Chih-Hai Su, Jie-Ying Lee, Shr-Ruei Tsai, Chin-Yang Lin, Kuan-Wen Chen, Tsung-Wei Ke, and Yu-Lun Liu. See, point, fly: A learning-free VLM framework for universal unmanned aerial navigation. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 4697–4708. PMLR, 2025.

[18] Zhenxing Xu, Brikit Lu, Weidong Bao, Zhengqiu Zhu, Junsong Zhang, Hui Yan, Wenhao Lu, and Ji Wang. Fly0: Decoupling semantic grounding from geometric planning for zero-shot aerial navigation, 2026.

[19] Xuecheng Chen, Zongzhuo Liu, Jianfa Ma, Bang Du, Tiantian Zhang, Xueqian Wang, and Boyu Zhou. Airhunt: Bridging VLM semantics and continuous planning for efficient aerial object navigation, 2026.

[20] Guiyong Zheng, Yueting Ban, Mingjie Zhang, Juepeng Zheng, and Boyu Zhou. Onfly: Onboard zero-shot aerial vision-language navigation toward safety and efficiency, 2026.

[21] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023.

[22] Marco S. Tayar, Lucas K. de Oliveira, Felipe Andrade G. Tommaselli, Juliano D. Negri, Thiago H. Segreto, Ricardo V. Godoy, and Marcelo Becker. Autonomous UAV flight navigation in confined spaces: A reinforcement learning approach. In *2025 Latin American Robotics Symposium (LARS)*, pages 1–6, 2025.

[23] Lucas K. de Oliveira, Felipe A. G. Tommaselli, João Aires Marsicano, Marco S. Tayar, Pedro A. R. Saraiva, Ricardo V. Godoy, and Marcelo Becker. Mira: A modular open-source micro-uav for indoor research, 2026.

http://www.imavs.org/

MRAC-Based Admittance Control for Aerial Physical Interaction

Ishaq Hafez* and Karl Stol
Department of Mechanical and Mechatronics Engineering, University of Auckland, Auckland 1010, New Zealand

http://www.imavs.org/

Abstract

This paper presents an admittance controller with model reference adaptive control (MRAC) for a fully actuated multirotor in physical interaction, in which an adaptive input drives the compliant response toward a reference compliance model. The admittance parameters are updated online through Lyapunov-based adaptation, while an MRAC position controller compensates matched translational uncertainty. Lyapunov analyses establish bounded tracking and parameter errors for the individual adaptive subsystems under the stated assumptions. Simulations on a fixed-tilt octocopter in single-point contact with a flexible cantilever beam show reduced interaction-axis tracking error and oscillations, increased beam-tip deformation, and greater robustness to parametric uncertainty than fixed admittance, with peak forces below the contact force limit.

1 Introduction

Aerial robots are increasingly employed in tasks requiring controlled physical interaction, including nondestructive inspection, infrastructure maintenance, and collaborative manipulation with humans [1, 2, 3]. Unlike fixed-base manipulators, unmanned aerial vehicles (UAVs) must regulate flight dynamics and contact forces simultaneously, which couples compliance, tracking accuracy, and contact safety. This creates a trade-off between compliance and precision: high compliance improves interaction safety but degrades free-flight tracking, whereas high stiffness improves positioning but can produce large impact forces and contact instability [4]. This trade-off motivates a control architecture that adapts compliant interaction and position tracking together under contact and model uncertainty.

Adaptive admittance control strategies have been developed for UAV physical interaction. Hamaza et al. [5] implemented an adaptive compliance mechanism on a quadrotor with a compliant arm by tuning torsional spring gains online to shape the force–position behavior during contact. For sustained contact in aerial manipulation, Marković et al. [6] used adaptive stiffness estimation within an impedance controller to regulate interaction with unknown environment stiffness. These approaches improve robustness to payload changes, unmodeled contact dynamics, and external disturbances, but adapt only the interaction layer. Simultaneous adaptation with motion tracking remains largely unexplored.

MRAC has addressed model uncertainty and external disturbances in UAV-manipulator systems. Orsag et al. used Lyapunov-based MRAC on a two-arm aerial manipulator to compensate center-of-mass and inertia variations, with a hybrid gain-scheduled extension [7, 8]. Sumathy and Ghose combined feedback linearization with MRAC to estimate uncertain parameters for a quadcopter with a 3-DoF arm [9]. Ali and Li developed an LQR-MRAC framework for a quadrotor with a 2-DoF gripper under payload-induced uncertainty [10]. Other studies address varying payloads and online estimation of mass, thrust, inertia, and center-of-gravity changes [11, 12]. These studies show reference-model adaptation compensates for payload changes, inertia shifts, and reaction forces, but does not adapt compliant behavior, such as force regulation, in coordination with trajectory tracking.

Aerial physical interaction requires simultaneous adaptation of compliant contact behavior and motion tracking under uncertainty. Prior works either employ an outer-loop MRAC with inner-loop controllers or use impedance or admittance control to shape interaction dynamics without MRAC position tracking, including adaptive stiffness and impedance control [6], interaction force and predictive admittance control on fully actuated platforms [1], port-Hamiltonian wrench and impedance regulation [13], and coordinated frameworks for compliant aerial manipulation [3]. Existing work does not combine MRAC-based input augmentation inside the admittance loop, Lyapunov-based admittance-parameter adaptation, and an MRAC position controller for adaptive aerial physical interaction. The proposed framework addresses this gap by adapting the contact response in the admittance layer and compensating matched translational uncertainty in the position loop. An MRAC term drives the admittance response toward a reference compliance model, while Lyapunov-based updates adapt the admittance parameters under contact and modeling variations.

Simulations evaluate the method on a fixed-tilt octocopter in contact with a flexible cantilever beam. A fixed-admittance baseline isolates the effect of online adaptation, while lateral and vertical contact-arm axes are considered to assess post-contact interaction-axis tracking error and oscillations under beam-model parametric uncertainty.

*Email address: ishaq.hafez@auckland.ac.nz

http://www.imavs.org/

2 System Model

The fixed-tilt octocopter (FT-X8) has eight rotors arranged in four fixed-tilt coaxial pairs [14], as shown in Fig. 1. This layout enables direct thrust along $\hat{\mathbf{x}}_b$ and $\hat{\mathbf{y}}_b$ without attitude changes and supports independent force and moment control. Contact is made through a body-mounted rigid contact arm whose axis is fixed in $\mathcal{F}_b$. The inertial frame $\mathcal{F}_n$ uses North-East-Down axes, and $\mathcal{F}_b$ is fixed at the center of mass with $\hat{\mathbf{z}}_b$ downward. Counter-rotating coaxial pairs cancel hover reaction torque, while differential thrust generates body-frame forces and moments. The attitude is parameterized by ZYX Euler angles $\boldsymbol{\eta} = [\phi, \theta, \psi]^\top$, where $\mathbf{R}_b^n$ maps $\mathcal{F}_b$ to $\mathcal{F}_n$ and $\mathbf{R}_n^b := (\mathbf{R}_b^n)^\top$. The body angular velocity $\boldsymbol{\omega}_b = [p,\ q,\ r]^\top$ satisfies $\dot{\boldsymbol{\eta}} = \mathbf{T}^{-1}(\phi, \theta)\boldsymbol{\omega}_b$ [12].

The translational dynamics in $\mathcal{F}_b$ are given by

$$m\big(\dot{\mathbf{v}}_b + \boldsymbol{\omega}_b \times \mathbf{v}_b\big) = mg\,\mathbf{R}_n^b \hat{\mathbf{z}}_n - \mathbf{C}_t\,\mathbf{v}_b + \mathbf{f}_b + \mathbf{f}_{e,b}, \quad (1)$$

where $m > 0$ is the mass, $\mathbf{v}_b := [\,v_{x,b}\ v_{y,b}\ v_{z,b}\,]^\top$ is the body-frame linear velocity, $g > 0$ is the gravitational acceleration, $\hat{\mathbf{z}}_n := [\,0\ 0\ 1\,]^\top$ is the inertial z-axis unit vector, $\mathbf{C}_t := \mathrm{diag}(C_{tx}, C_{ty}, C_{tz})$ is the translational drag matrix, $\mathbf{f}_b := [\,f_{x,b}\ f_{y,b}\ f_{z,b}\,]^\top$ is the total rotor thrust force, and $\mathbf{f}_{e,b} \in \mathbb{R}^3$ is the external contact force. Let $\mathbf{p}_n := [\,x_n\ y_n\ z_n\,]^\top$ denote the position in $\mathcal{F}_n$. Differentiating $\dot{\mathbf{p}}_n = \mathbf{R}_b^n\,\mathbf{v}_b$ and using (1) gives the inertial acceleration

$$\ddot{\mathbf{p}}_n = g\,\hat{\mathbf{z}}_n + \tfrac{1}{m}\,\mathbf{R}_b^n\big(\mathbf{f}_b - \mathbf{C}_t\,\mathbf{v}_b + \mathbf{f}_{e,b}\big). \quad (2)$$

The rotational dynamics in $\mathcal{F}_b$ are given by

$$\mathbf{J}\,\dot{\boldsymbol{\omega}}_b + \boldsymbol{\omega}_b \times (\mathbf{J}\,\boldsymbol{\omega}_b) = -\,\mathbf{C}_r\,\boldsymbol{\omega}_b + \mathbf{m}_b + \boldsymbol{\tau}_{e,b}. \quad (3)$$

where $\mathbf{J} := \mathrm{diag}(J_x, J_y, J_z)$ is the body inertia matrix, $\mathbf{C}_r := \mathrm{diag}(c_p, c_q, c_r)$ is the rotational damping matrix, $\mathbf{m}_b := [m_{x,b}\ m_{y,b}\ m_{z,b}]^\top$ is the rotor moment input, and $\boldsymbol{\tau}_{e,b} := [\tau_{e,x,b}\ \tau_{e,y,b}\ \tau_{e,z,b}]^\top$ is the external contact moment.

For rotor $i \in \{1, \ldots, 8\}$, let $\mathcal{F}_i$ be its local frame, $\hat{\mathbf{z}}_i$ its thrust axis, $\mathbf{R}_b^i$ the rotation from $\mathcal{F}_i$ to $\mathcal{F}_b$, and $\mathbf{p}_i := [\,x_i\ y_i\ z_i\,]^\top$ its position in $\mathcal{F}_b$. The total body wrench generated by the rotor thrusts and reaction torques is

$$\mathbf{w}_b := \begin{bmatrix}\mathbf{f}_b \\ \mathbf{m}_b\end{bmatrix} = \sum_{i=1}^{8} \begin{bmatrix}\mathbf{R}_b^i\big(f_i\,\hat{\mathbf{z}}_i\big) \\ \mathbf{R}_b^i\big(m_i\,\hat{\mathbf{z}}_i\big) + \mathbf{p}_i \times \mathbf{R}_b^i\big(f_i\,\hat{\mathbf{z}}_i\big)\end{bmatrix}. \quad (4)$$

The dynamics use $\mathbf{w}_b + \mathbf{w}_{e,b}$, where $\mathbf{w}_{e,b} := [\,\mathbf{f}_{e,b}^\top\ \boldsymbol{\tau}_{e,b}^\top\,]^\top$, with $\mathbf{f}_{e,b}$ and $\boldsymbol{\tau}_{e,b}$ entering (1) and (3), respectively. Each rotor follows $f_i = k_f\omega_i^2$ and $m_i = k_m\omega_i^2$ for $i \in \{1, \ldots, 8\}$, where ω_i is the rotor speed, and $k_f > 0$ and $k_m > 0$ are the thrust and drag torque coefficients. Coaxial rotor-wash effects are modeled approximately by assigning the lower rotors in each coaxial pair an effectiveness factor of 0.8. Collecting the FT-X8 rotor thrusts as $\mathbf{f}_r := [\,f_1\ \cdots\ f_8\,]^\top \in \mathbb{R}^8$ with fixed tilt angle β, (4) becomes $\mathbf{w}_b = \boldsymbol{\Lambda}\mathbf{f}_r$, where $\boldsymbol{\Lambda} \in \mathbb{R}^{6\times 8}$ is

$$\boldsymbol{\Lambda} = \begin{bmatrix} -s_\beta & -s_\beta & s_\beta & s_\beta & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & s_\beta & -s_\beta & -s_\beta & s_\beta \\ -c_\beta & -c_\beta & -c_\beta & -c_\beta & -c_\beta & -c_\beta & -c_\beta & -c_\beta \\ -\sigma_2 & \sigma_2 & \sigma_2 & -\sigma_2 & \sigma_1 & -\sigma_1 & -\sigma_1 & \sigma_1 \\ \sigma_1 & \sigma_1 & -\sigma_1 & -\sigma_1 & \sigma_2 & \sigma_2 & -\sigma_2 & -\sigma_2 \\ \sigma_3 & -\sigma_3 & \sigma_3 & -\sigma_3 & \sigma_3 & -\sigma_3 & \sigma_3 & -\sigma_3 \end{bmatrix},$$

where $c_\beta = \cos\beta$, $s_\beta = \sin\beta$, $\sigma_1 = \frac{\sqrt{2}}{4}D\cos\beta$, $\sigma_2 = \sigma_1 - \frac{k_m}{k_f}\sin\beta$, and $\sigma_3 = \frac{k_m}{k_f}\cos\beta + \frac{\sqrt{2}}{4}D\sin\beta$. Since $\mathrm{rank}(\boldsymbol{\Lambda}) = 6$ for $\beta \in (0, \frac{\pi}{2})$, the platform is fully actuated at the body-wrench level. Although the FT-X8 is overactuated, the controller design does not rely on rotor redundancy.

Let $\mathbf{u}_{b,\mathrm{cmd}} := [\,\mathbf{f}_{b,\mathrm{cmd}}^\top\ \mathbf{m}_{b,\mathrm{cmd}}^\top\,]^\top \in \mathbb{R}^6$ denote the commanded body wrench and $\mathbf{f}_{r,\mathrm{cmd}} := [\,f_{\mathrm{cmd},1}\ \cdots\ f_{\mathrm{cmd},8}\,]^\top \in \mathbb{R}^8$ the commanded rotor thrust vector. The allocator maps $\mathbf{u}_{b,\mathrm{cmd}}$ to feasible rotor thrusts using $\boldsymbol{\Lambda}$ and sets $\omega_{i,\mathrm{cmd}} = \sqrt{f_{\mathrm{cmd},i}/k_f}$. Motor dynamics are modeled as $\tau\dot{\omega}_i = \omega_{i,\mathrm{cmd}} - \omega_i$, where $\tau > 0$ is the motor time constant.

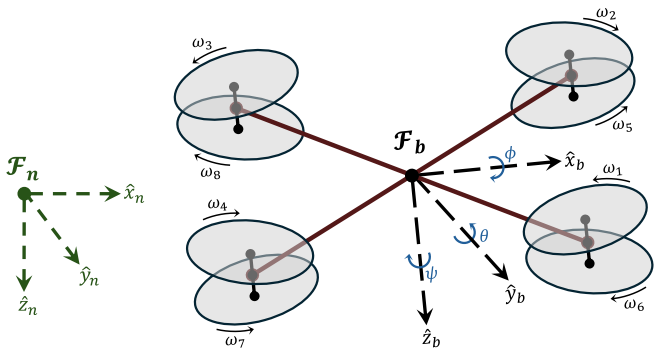


Figure 1: FT-X8 rotor layout with inertial and body frames.

3 Control Architecture

The control architecture in Fig. 2 maps the reference position and yaw command $\mathbf{r}_q(t) := [\,r_{x,n}(t)\ r_{y,n}(t)\ r_{z,n}(t)\ r_\psi(t)\,]^\top$ and the external wrench estimate $\hat{\mathbf{w}}_{e,n}(t) := [\,\hat{f}_{x,n}(t)\ \hat{f}_{y,n}(t)\ \hat{f}_{z,n}(t)\ \hat{m}_\psi(t)\,]^\top$ to compliant setpoints $\mathbf{q}_{sp}(t) := [\,x_{sp}(t)\ y_{sp}(t)\ z_{sp}(t)\ \psi_{sp}(t)\,]^\top$. The adaptive admittance layer regulates compliant interaction using an MRAC input and online parameter updates, while the MRAC position controller uses the translational components of $\mathbf{q}_{sp}(t)$ to generate $\boldsymbol{a}_{\mathrm{cmd}}(t) := [\,a_{x,n}(t)\ a_{y,n}(t)\ a_{z,n}(t)\,]^\top$ and compensate matched translational uncertainty. The inner flight controller and control allocation drive the plant to track $\boldsymbol{a}_{\mathrm{cmd}}(t)$ using cascaded attitude and rate PID loops [15].

3.1 Fixed Admittance Controller

The fixed admittance (FA) controller imposes a linear mass–spring–damper relation between wrench and compliant setpoint for $q \in \{x_n, y_n, z_n, \psi\}$. Let $r_{q,n}$, $q_{sp,n}$, and $\hat{f}_{q,n}$

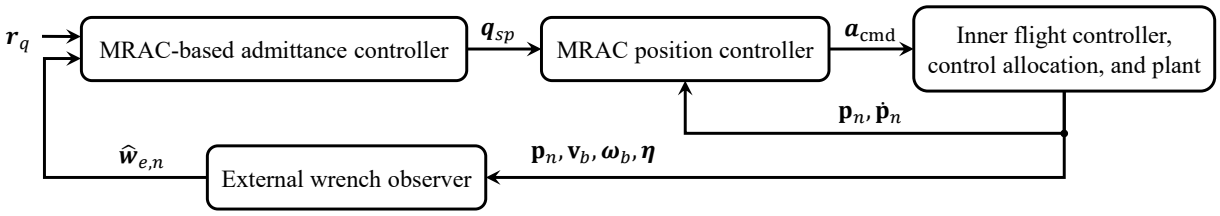


Figure 2: MRAC-based control architecture for aerial physical interaction.

denote the reference, compliant setpoint, and wrench component, and define $e_q^{\mathrm{c}}(t) := q_{sp,n}(t) - r_{q,n}(t)$. An observer [16] provides $\hat{\mathbf{w}}_{e,n}(t) \in \mathbb{R}^4$ from measured $(\mathbf{p}_n, \mathbf{v}_b, \boldsymbol{\omega}_b, \boldsymbol{\eta})$, while the plant is driven by the contact wrench $\mathbf{w}_{e,b}(t)$. The compliant setpoint follows a second-order admittance law [2].

$$\begin{aligned} &M_q\big(\ddot{q}_{sp,n}(t) - \ddot{r}_{q,n}(t)\big) + C_q\big(\dot{q}_{sp,n}(t) - \dot{r}_{q,n}(t)\big) \\ &\quad + K_q\big(q_{sp,n}(t) - r_{q,n}(t)\big) = \hat{f}_{q,n}(t). \end{aligned} \tag{5}$$

with $M_q \in \mathbb{R}_{>0}$, $C_q \in \mathbb{R}_{>0}$, and $K_q \in \mathbb{R}_{>0}$. Equation (5) defines four decoupled second-order admittance filters on x_n, y_n, z_n, and ψ. Roll and pitch are stabilized by the inner loop and excluded from the admittance law.

3.2 MRAC-Based Adaptive Admittance Control

The FA law in (5) is sensitive to contact uncertainty, wrench estimation errors, unmodeled couplings, and fixed admittance parameters, which can cause residual offsets in $q_{sp,n} - r_{q,n}$. The proposed MRAC-based admittance uses $u_q(t)$ to drive $q_{sp,n}$ toward a reference model, while online parameter updates adapt the compliance dynamics. Introducing $u_q(t)$ gives the channel-q admittance dynamics

$$M_q \ddot{e}_q^{\mathrm{c}}(t) + C_q \dot{e}_q^{\mathrm{c}}(t) + K_q e_q^{\mathrm{c}}(t) = \hat{f}_{q,n}(t) + u_q(t). \tag{6}$$

Let $\mathbf{x}_q(t) := [\, q_{sp,n}(t) \;\; \dot{q}_{sp,n}(t) \,]^\top$. Equation (6) becomes

$$\dot{\mathbf{x}}_q(t) = \mathbf{A}_q\, \mathbf{x}_q(t) + \mathbf{B}_q\big(u_q(t) + \nu_q(t) + d_q\big),$$

$$\mathbf{A}_q := \begin{bmatrix} 0 & 1 \\ -\frac{K_q}{M_q} & -\frac{C_q}{M_q} \end{bmatrix}, \qquad \mathbf{B}_q := \begin{bmatrix} 0 \\ \frac{1}{M_q} \end{bmatrix},$$

where $d_q \in \mathbb{R}$ is an unknown constant matched disturbance representing wrench-estimation bias and unmodeled contact terms, and

$$\nu_q(t) := \hat{f}_{q,n}(t) + M_q \ddot{r}_{q,n}(t) + C_q \dot{r}_{q,n}(t) + K_q r_{q,n}(t)$$

is a known matched term. The desired reference model is

$$\dot{\mathbf{x}}_{\mathrm{ref},q}(t) = \mathbf{A}_{\mathrm{ref},q}\, \mathbf{x}_{\mathrm{ref},q}(t) + \mathbf{B}_{\mathrm{ref},q}\, r_{q,n}(t),$$

with $\mathbf{x}_{\mathrm{ref},q}(t) := [\, q_{\mathrm{ref},q}(t) \;\; \dot{q}_{\mathrm{ref},q}(t) \,]^\top$ and

$$\mathbf{A}_{\mathrm{ref},q} := \begin{bmatrix} 0 & 1 \\ -\omega_{n,q}^2 & -2\zeta_q\omega_{n,q} \end{bmatrix}, \quad \mathbf{B}_{\mathrm{ref},q} := \begin{bmatrix} 0 \\ \omega_{n,q}^2 \end{bmatrix},$$

where $\zeta_q > 0$ and $\omega_{n,q} > 0$, with settling time $T_{\mathrm{set},q} \approx 4/(\zeta_q\omega_{n,q})$. Define the admittance MRAC tracking error

$$\mathbf{e}_q^{\mathrm{m}}(t) := \mathbf{x}_q(t) - \mathbf{x}_{\mathrm{ref},q}(t) = \begin{bmatrix} q_{sp,n}(t) - q_{\mathrm{ref},q}(t) \\ \dot{q}_{sp,n}(t) - \dot{q}_{\mathrm{ref},q}(t) \end{bmatrix}.$$

The adaptive input is chosen in the direct MRAC form [17],

$$\begin{aligned} u_q(t) = \hat{K}_{q1}(t)\, q_{sp,n}(t) + \hat{K}_{q2}(t)\, \dot{q}_{sp,n}(t) \\ + \hat{K}_{qr}(t)\, r_{q,n}(t) - \nu_q(t) - \hat{K}_{qd}(t), \end{aligned}$$

where $\hat{K}_{q1}(t)$, $\hat{K}_{q2}(t)$, $\hat{K}_{qr}(t)$, and $\hat{K}_{qd}(t)$ are adaptive estimates. The term $-\nu_q(t)$ cancels the known matched term, and $\hat{K}_{qd}(t)$ compensates d_q. Assume ideal matching constants $K_{q1}^\star, K_{q2}^\star, K_{qr}^\star, K_{qd}^\star$ exist such that

$$\begin{aligned} \mathbf{A}_q + \mathbf{B}_q \begin{bmatrix} K_{q1}^\star & K_{q2}^\star \end{bmatrix} &= \mathbf{A}_{\mathrm{ref},q}, \\ \mathbf{B}_q K_{qr}^\star &= \mathbf{B}_{\mathrm{ref},q}, \\ K_{qd}^\star &= d_q. \end{aligned}$$

Define the parameter errors

$$\tilde{K}_{qi}(t) := \hat{K}_{qi}(t) - K_{qi}^\star, \qquad i \in \{1, 2, r, d\}.$$

Then the admittance MRAC error dynamics become

$$\begin{aligned} \dot{\mathbf{e}}_q^{\mathrm{m}}(t) = \mathbf{A}_{\mathrm{ref},q}\mathbf{e}_q^{\mathrm{m}}(t) + \mathbf{B}_q\Big(&\tilde{K}_{q1}(t) q_{sp,n}(t) \\ &+ \tilde{K}_{q2}(t)\dot{q}_{sp,n}(t) + \tilde{K}_{qr}(t) r_{q,n}(t) - \tilde{K}_{qd}(t)\Big). \end{aligned} \tag{7}$$

Let $\mathbf{P}_q = \mathbf{P}_q^\top \succ 0$ solve

$$\mathbf{P}_q\mathbf{A}_{\mathrm{ref},q} + \mathbf{A}_{\mathrm{ref},q}^\top\mathbf{P}_q = -\mathbf{Q}_q, \qquad \mathbf{Q}_q = \mathbf{Q}_q^\top \succ 0,$$

and define $\sigma_q(t) := \mathbf{e}_q^{\mathrm{m}\top}(t)\mathbf{P}_q\mathbf{B}_q$. The adaptive laws are

$$\dot{\hat{K}}_{q1}(t) = -\Gamma_{q1}\, q_{sp,n}(t)\, \sigma_q(t), \; \dot{\hat{K}}_{q2}(t) = -\Gamma_{q2}\, \dot{q}_{sp,n}(t)\, \sigma_q(t),$$
$$\dot{\hat{K}}_{qr}(t) = -\Gamma_{qr}\, r_{q,n}(t)\, \sigma_q(t), \; \dot{\hat{K}}_{qd}(t) = \Gamma_{qd}\, \sigma_q(t),$$

with $\Gamma_{q1}, \Gamma_{q2}, \Gamma_{qr}, \Gamma_{qd} \in \mathbb{R}_{>0}$. For the Lyapunov function

$$\begin{aligned} V_q^{\mathrm{m}}(t) = \mathbf{e}_q^{\mathrm{m}\top}(t)\mathbf{P}_q\mathbf{e}_q^{\mathrm{m}}(t) + \frac{\tilde{K}_{q1}^2(t)}{\Gamma_{q1}} + \frac{\tilde{K}_{q2}^2(t)}{\Gamma_{q2}} \\ + \frac{\tilde{K}_{qr}^2(t)}{\Gamma_{qr}} + \frac{\tilde{K}_{qd}^2(t)}{\Gamma_{qd}}, \end{aligned}$$

http://www.imavs.org/

using (7) and the adaptive laws gives the derivative $\dot{V}_q^{\mathrm{m}}(t) = -\mathbf{e}_q^{\mathrm{m}\top}(t)\mathbf{Q}_q\mathbf{e}_q^{\mathrm{m}}(t) \le 0$. Thus, $\mathbf{e}_q^{\mathrm{m}}(t)$ and the parameter errors are bounded. Under standard MRAC conditions, $\mathbf{e}_q^{\mathrm{m}}(t) \to 0$.

The parameters (M_q, C_q, K_q) are updated online. Define

$$\hat{a}_{1,q}(t) := \frac{\hat{C}_q(t)}{\hat{M}_q(t)}, \quad \hat{a}_{0,q}(t) := \frac{\hat{K}_q(t)}{\hat{M}_q(t)}, \quad \hat{b}_q(t) := \frac{1}{\hat{M}_q(t)}.$$

Substituting into the admittance error dynamics (5) gives

$$\ddot{e}_q^{\mathrm{c}}(t) = -\hat{a}_{1,q}(t)\dot{e}_q^{\mathrm{c}}(t) - \hat{a}_{0,q}(t)e_q^{\mathrm{c}}(t) + \hat{b}_q(t)\hat{f}_{q,n}(t). \quad (8)$$

Let $\mathbf{x}_q^{\mathrm{c}}(t) := [\, e_q^{\mathrm{c}}(t) \;\; \dot{e}_q^{\mathrm{c}}(t)\,]^\top$ and define

$$\hat{\mathbf{A}}_q^{\mathrm{c}}(t) := \begin{bmatrix} 0 & 1 \\ -\hat{a}_{0,q}(t) & -\hat{a}_{1,q}(t) \end{bmatrix}, \qquad \mathbf{B}_c := \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Then the compliance error dynamics (8) becomes

$$\dot{\mathbf{x}}_q^{\mathrm{c}}(t) = \hat{\mathbf{A}}_q^{\mathrm{c}}(t)\,\mathbf{x}_q^{\mathrm{c}}(t) + \mathbf{B}_c\,\hat{b}_q(t)\hat{f}_{q,n}(t).$$

The ideal compliance model is specified by $a_{0,q}^\star = \omega_{n,q}^2$, $a_{1,q}^\star = 2\zeta_q\omega_{n,q}$, and $b_q^\star > 0$, with

$$\mathbf{A}_q^{\mathrm{c}\star} := \begin{bmatrix} 0 & 1 \\ -a_{0,q}^\star & -a_{1,q}^\star \end{bmatrix}.$$

Let $\mathbf{x}_{m,q}(t)$ be the state of the corresponding reference model

$$\dot{\mathbf{x}}_{m,q}(t) = \mathbf{A}_q^{\mathrm{c}\star}\,\mathbf{x}_{m,q}(t) + \mathbf{B}_c\,b_q^\star\hat{f}_{q,n}(t),$$

and define the compliance-model tracking error $\mathbf{e}_q^{\mathrm{cm}}(t) := \mathbf{x}_q^{\mathrm{c}}(t) - \mathbf{x}_{m,q}(t)$. The error dynamics are given by

$$\dot{\mathbf{e}}_q^{\mathrm{cm}}(t) = \mathbf{A}_q^{\mathrm{c}\star}\,\mathbf{e}_q^{\mathrm{cm}}(t) + \mathbf{B}_c\big(-\tilde{a}_{1,q}\dot{e}_q^{\mathrm{c}}(t) - \tilde{a}_{0,q}e_q^{\mathrm{c}}(t) + \tilde{b}_q\hat{f}_{q,n}(t)\big),$$

with $\tilde{a}_{1,q}(t) := \hat{a}_{1,q}(t) - a_{1,q}^\star$, $\tilde{a}_{0,q}(t) := \hat{a}_{0,q}(t) - a_{0,q}^\star$, and $\tilde{b}_q(t) := \hat{b}_q(t) - b_q^\star$. Let $\mathbf{P}_q^{\mathrm{c}} = \mathbf{P}_q^{\mathrm{c}\top} \succ 0$ solve

$$\mathbf{A}_q^{\mathrm{c}\star\top}\mathbf{P}_q^{\mathrm{c}} + \mathbf{P}_q^{\mathrm{c}}\mathbf{A}_q^{\mathrm{c}\star} = -\mathbf{Q}_q^{\mathrm{c}}, \qquad \mathbf{Q}_q^{\mathrm{c}} = \mathbf{Q}_q^{\mathrm{c}\top} \succ 0,$$

and define $\sigma_q^{\mathrm{c}}(t) := \mathbf{e}_q^{\mathrm{cm}\top}(t)\mathbf{P}_q^{\mathrm{c}}\mathbf{B}_c$. For $\Gamma_{a1,q}, \Gamma_{a0,q}, \Gamma_{b,q} \in \mathbb{R}_{>0}$, consider the Lyapunov function

$$V_q^{\mathrm{c}} = \mathbf{e}_q^{\mathrm{cm}\top}(t)\mathbf{P}_q^{\mathrm{c}}\mathbf{e}_q^{\mathrm{cm}}(t) + \frac{\tilde{a}_{1,q}^2}{\Gamma_{a1,q}} + \frac{\tilde{a}_{0,q}^2}{\Gamma_{a0,q}} + \frac{\tilde{b}_q^2}{\Gamma_{b,q}}.$$

Choosing

$$\begin{aligned} \dot{\hat{a}}_{1,q}(t) &= \Gamma_{a1,q}\,\dot{e}_q^{\mathrm{c}}(t)\,\sigma_q^{\mathrm{c}}(t), \\ \dot{\hat{a}}_{0,q}(t) &= \Gamma_{a0,q}\,e_q^{\mathrm{c}}(t)\,\sigma_q^{\mathrm{c}}(t), \\ \dot{\hat{b}}_q(t) &= -\Gamma_{b,q}\,\hat{f}_{q,n}(t)\,\sigma_q^{\mathrm{c}}(t), \end{aligned}$$

yields the derivative $\dot{V}_q^{\mathrm{c}}(t) = -\mathbf{e}_q^{\mathrm{cm}\top}(t)\mathbf{Q}_q^{\mathrm{c}}\mathbf{e}_q^{\mathrm{cm}}(t) \le 0$, so $\mathbf{e}_q^{\mathrm{cm}}(t)$, $\tilde{a}_{1,q}$, $\tilde{a}_{0,q}$, and $\tilde{b}_q$ are bounded and $\mathbf{e}_q^{\mathrm{cm}}(t) \to 0$ as $t \to \infty$. The adaptive admittance parameters are then

$$\hat{M}_q(t) = \frac{1}{\hat{b}_q(t)}, \; \hat{C}_q(t) = \frac{\hat{a}_{1,q}(t)}{\hat{b}_q(t)}, \; \hat{K}_q(t) = \frac{\hat{a}_{0,q}(t)}{\hat{b}_q(t)}.$$

The adaptive admittance for each channel is therefore

$$\hat{M}_q(t)\,\ddot{e}_q^{\mathrm{c}}(t) + \hat{C}_q(t)\,\dot{e}_q^{\mathrm{c}}(t) + \hat{K}_q(t)\,e_q^{\mathrm{c}}(t) = \hat{f}_{q,n}(t) + u_q(t),$$

where $u_q(t)$ drives $q_{sp,n}(t)$ toward the reference model, and $(\hat{M}_q, \hat{C}_q, \hat{K}_q)$ define the adaptive admittance dynamics.

3.3 MRAC-Based Position Control

The MRAC position controller tracks the translational components of $\mathbf{q}_{sp}(t)$ by commanding inertial accelerations on $q \in \{x_n, y_n, z_n\}$. From (2), with $\mathbf{x}_q(t) := [\, q_n(t) \;\; \dot{q}_n(t)\,]^\top$ and $u_q^{\mathrm{p}}(t) := a_{n,q,\mathrm{cmd}}(t)$, the matched-uncertainty model is

$$\dot{\mathbf{x}}_q(t) = \mathbf{A}\,\mathbf{x}_q(t) + \mathbf{B}\big(u_q^{\mathrm{p}}(t) + \boldsymbol{\Theta}_q^\top\boldsymbol{\Phi}_q(t) + d_q\big),$$

with

$$\mathbf{A} := \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \qquad \mathbf{B} := \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

where $d_q \in \mathbb{R}$ is an unknown constant matched disturbance, $\boldsymbol{\Theta}_q \in \mathbb{R}^{p_q}$ is an unknown parameter vector, and $\boldsymbol{\Phi}_q(t) \in \mathbb{R}^{p_q}$ is a known regressor. The pair $(\boldsymbol{\Theta}_q, d_q)$ represents unmodeled acceleration terms. From (2), define $\boldsymbol{\rho}_q^\top(t) := \mathbf{i}_q^\top\mathbf{R}_b^n(t)$, where $\mathbf{i}_q$ is the unit vector along $q \in \{x_n, y_n, z_n\}$. Then

$$\boldsymbol{\Phi}_q(t) := \begin{bmatrix} \rho_{q1}(t)f_{x,b}(t) \\ \rho_{q2}(t)f_{y,b}(t) \\ \rho_{q3}(t)f_{z,b}(t) \end{bmatrix}, \qquad p_q = 3.$$

Choose a second-order reference model,

$$\dot{\mathbf{x}}_{\mathrm{ref},q}^{\mathrm{p}}(t) = \mathbf{A}_{\mathrm{ref},q}^{\mathrm{p}}\,\mathbf{x}_{\mathrm{ref},q}^{\mathrm{p}}(t) + \mathbf{B}_{\mathrm{ref},q}^{\mathrm{p}}\,q_{sp,n}(t),$$

where $\mathbf{x}_{\mathrm{ref},q}^{\mathrm{p}}(t) := [\, q_{\mathrm{ref},q}^{\mathrm{p}}(t) \;\; \dot{q}_{\mathrm{ref},q}^{\mathrm{p}}(t)\,]^\top$, and

$$\mathbf{A}_{\mathrm{ref},q}^{\mathrm{p}} := \begin{bmatrix} 0 & 1 \\ -\omega_{p,q}^2 & -2\zeta_{p,q}\omega_{p,q} \end{bmatrix}, \qquad \mathbf{B}_{\mathrm{ref},q}^{\mathrm{p}} := \begin{bmatrix} 0 \\ \omega_{p,q}^2 \end{bmatrix}.$$

The parameters $\omega_{p,q} > 0$ and $\zeta_{p,q} > 0$ tune the desired position transient. The position MRAC tracking error is

$$\mathbf{e}_q^{\mathrm{p}}(t) := \mathbf{x}_q(t) - \mathbf{x}_{\mathrm{ref},q}^{\mathrm{p}}(t) = \begin{bmatrix} q_n(t) - q_{\mathrm{ref},q}^{\mathrm{p}}(t) \\ \dot{q}_n(t) - \dot{q}_{\mathrm{ref},q}^{\mathrm{p}}(t) \end{bmatrix}.$$

The position control law on axis q is chosen as

$$\begin{aligned} u_q^{\mathrm{p}}(t) = \hat{K}_{x,q}^\top(t)\,\mathbf{x}_q(t) + \hat{K}_{r,q}(t)\,q_{sp,n}(t) \\ - \hat{\boldsymbol{\Theta}}_q^\top(t)\,\boldsymbol{\Phi}_q(t) - \hat{K}_{d,q}(t), \end{aligned}$$

with adaptive parameters $\hat{K}_{x,q}(t) \in \mathbb{R}^2$, $\hat{K}_{r,q}(t) \in \mathbb{R}$, $\hat{\boldsymbol{\Theta}}_q(t) \in \mathbb{R}^{p_q}$, and $\hat{K}_{d,q}(t) \in \mathbb{R}$. Assume ideal constants $K_{x,q}^\star \in \mathbb{R}^2$, $K_{r,q}^\star \in \mathbb{R}$, $\boldsymbol{\Theta}_q^\star \in \mathbb{R}^{p_q}$, and $K_{d,q}^\star \in \mathbb{R}$ such that

$$\begin{aligned} \mathbf{A} + \mathbf{B}K_{x,q}^{\star\top} = \mathbf{A}_{\mathrm{ref},q}^{\mathrm{p}}, \quad \mathbf{B}K_{r,q}^\star = \mathbf{B}_{\mathrm{ref},q}^{\mathrm{p}}, \\ K_{d,q}^\star = d_q, \qquad \boldsymbol{\Theta}_q^\star = \boldsymbol{\Theta}_q. \end{aligned}$$

http://www.imavs.org/

http://www.imavs.org/

Define the position MRAC parameter errors

$$\tilde{K}_{x,q}(t) := \hat{K}_{x,q}(t) - K^{\star}_{x,q}, \quad \tilde{K}_{r,q}(t) := \hat{K}_{r,q}(t) - K^{\star}_{r,q},$$
$$\tilde{\mathbf{\Theta}}_q(t) := \hat{\mathbf{\Theta}}_q(t) - \mathbf{\Theta}^{\star}_q, \quad \tilde{K}_{d,q}(t) := \hat{K}_{d,q}(t) - K^{\star}_{d,q}.$$

Then the error dynamics for the position controller are

$$\dot{\mathbf{e}}^{\mathrm{p}}_q(t) = \mathbf{A}^{\mathrm{p}}_{\mathrm{ref},q}\,\mathbf{e}^{\mathrm{p}}_q(t) + \mathbf{B}\Big(\tilde{K}^{\top}_{x,q}(t)\,\mathbf{x}_q(t) + \tilde{K}_{r,q}(t)\,q_{sp,n}(t) - \tilde{\mathbf{\Theta}}^{\top}_q(t)\,\mathbf{\Phi}_q(t) - \tilde{K}_{d,q}(t)\Big).$$

Let $\mathbf{P}^{\mathrm{p}}_q = \mathbf{P}^{\mathrm{p}\top}_q \succ 0$ solve

$$\mathbf{P}^{\mathrm{p}}_q\mathbf{A}^{\mathrm{p}}_{\mathrm{ref},q} + \mathbf{A}^{\mathrm{p}\top}_{\mathrm{ref},q}\mathbf{P}^{\mathrm{p}}_q = -\mathbf{Q}^{\mathrm{p}}_q, \qquad \mathbf{Q}^{\mathrm{p}}_q = \mathbf{Q}^{\mathrm{p}\top}_q \succ 0.$$

Define $\sigma^{\mathrm{p}}_q(t) := \mathbf{e}^{\mathrm{p}\top}_q(t)\mathbf{P}^{\mathrm{p}}_q\mathbf{B}$. Use the Lyapunov function

$$V^{\mathrm{p}}_q(t) = \mathbf{e}^{\mathrm{p}\top}_q(t)\,\mathbf{P}^{\mathrm{p}}_q\,\mathbf{e}^{\mathrm{p}}_q(t) + \tilde{K}^{\top}_{x,q}(t)\,\mathbf{\Gamma}^{-1}_{x,q}\,\tilde{K}_{x,q}(t) + \frac{\tilde{K}^2_{r,q}(t)}{\Gamma_{r,q}} + \tilde{\mathbf{\Theta}}^{\top}_q(t)\,\mathbf{\Gamma}^{-1}_{\Theta,q}\,\tilde{\mathbf{\Theta}}_q(t) + \frac{\tilde{K}^2_{d,q}(t)}{\Gamma_{d,q}},$$

with $\mathbf{\Gamma}_{x,q}, \mathbf{\Gamma}_{\Theta,q} \succ 0$ and $\Gamma_{r,q}, \Gamma_{d,q} > 0$. Choosing

$$\dot{\hat{K}}_{x,q}(t) = -\mathbf{\Gamma}_{x,q}\,\mathbf{x}_q(t)\,\sigma^{\mathrm{p}}_q(t),$$
$$\dot{\hat{K}}_{r,q}(t) = -\Gamma_{r,q}\,q_{sp,n}(t)\,\sigma^{\mathrm{p}}_q(t),$$
$$\dot{\hat{\mathbf{\Theta}}}_q(t) = \mathbf{\Gamma}_{\Theta,q}\,\mathbf{\Phi}_q(t)\,\sigma^{\mathrm{p}}_q(t),$$
$$\dot{\hat{K}}_{d,q}(t) = \Gamma_{d,q}\,\sigma^{\mathrm{p}}_q(t),$$

yields the derivative $\dot{V}^{\mathrm{p}}_q(t) = -\mathbf{e}^{\mathrm{p}\top}_q(t)\mathbf{Q}^{\mathrm{p}}_q\mathbf{e}^{\mathrm{p}}_q(t) \le 0$. Therefore, $\mathbf{e}^{\mathrm{p}}_q(t)$ and the parameter errors are bounded. Under standard MRAC conditions, $\mathbf{e}^{\mathrm{p}}_q(t) \to 0$, provided the inner loop and allocation track within actuator limits.

4 RESULTS

The setup uses a body-mounted rigid contact arm whose tip contacts the free tip of a wall-mounted cantilever beam, as shown in Fig. 3. The beam acts as a compliant environment and generates an external wrench through the arm-tip reaction force. The beam centerline is aligned with $\hat{\mathbf{x}}_n := [1\ \ 0\ \ 0]^{\top}$ and has length $L_{\mathcal{B}}$. The UAV starts at $\mathbf{p}_n(0) = [0\ \ 0\ \ 0]^{\top}$ and tracks the undeformed beam-tip reference $\mathbf{p}_{T0,n} := [1\ \ 1\ \ -1]^{\top}$ over $T_{\mathrm{arr}} = 10$ s. The beam base is fixed at $\mathbf{p}_{0,n} := \mathbf{p}_{T0,n} - L_{\mathcal{B}}\,\hat{\mathbf{x}}_n$. This setup evaluates free-flight and contact operation without controller switching.

Let $\hat{\mathbf{a}}_b \in \mathbb{R}^3$ be the unit contact-arm axis fixed in $\mathcal{F}_b$, $s(t) \in [0, s_{\max}]$ the prescribed tip offset along this axis, and $\mathbf{r}_{a,b} \in \mathbb{R}^3$ the arm base. The arm-tip position and velocity are

$$\mathbf{r}_{c,b}(t) = \mathbf{r}_{a,b} + s(t)\,\hat{\mathbf{a}}_b, \qquad \dot{\mathbf{r}}_{c,b}(t) = \dot{s}(t)\,\hat{\mathbf{a}}_b.$$

The contact position and velocity in $\mathcal{F}_n$ are

$$\mathbf{p}_{c,n}(t) = \mathbf{p}_n(t) + \mathbf{R}^n_b(t)\,\mathbf{r}_{c,b}(t),$$
$$\mathbf{v}_{c,n}(t) = \dot{\mathbf{p}}_n(t) + \mathbf{R}^n_b(t)\big(\boldsymbol{\omega}_b(t) \times \mathbf{r}_{c,b}(t) + \dot{\mathbf{r}}_{c,b}(t)\big).$$

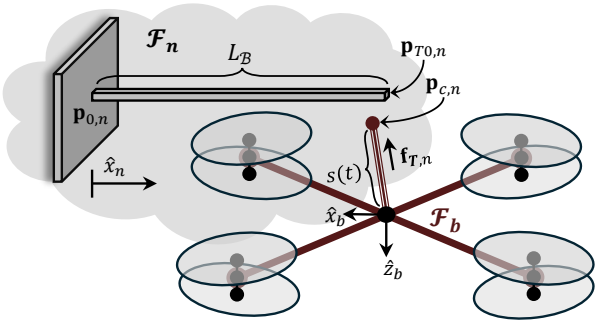


Figure 3: y_b-axis contact arm approaching a cantilever beam.

The beam-tip deformation in $\mathcal{F}_n$ is described by $\boldsymbol{\xi}(t) \in \mathbb{R}^3$,

$$\mathbf{p}_{T,n}(t) = \mathbf{p}_{T0,n} + \boldsymbol{\xi}(t), \qquad \mathbf{v}_{T,n}(t) = \dot{\boldsymbol{\xi}}(t).$$

The tip compliance is modeled by decoupled mass–spring–damper dynamics along the three inertial-frame axes:

$$\mathbf{M}_{\mathcal{B}}\,\ddot{\boldsymbol{\xi}}(t) + \mathbf{C}_{\mathcal{B}}\,\dot{\boldsymbol{\xi}}(t) + \mathbf{K}_{\mathcal{B}}\,\boldsymbol{\xi}(t) = \mathbf{f}_{T,n}(t),$$

where $\mathbf{M}_{\mathcal{B}}, \mathbf{C}_{\mathcal{B}}, \mathbf{K}_{\mathcal{B}} \in \mathbb{R}^{3\times3}$ are mass, damping, and stiffness with $\mathbf{M}_{\mathcal{B}} \succ 0$, $\mathbf{C}_{\mathcal{B}} \succeq 0$, and $\mathbf{K}_{\mathcal{B}} \succeq 0$, and $\mathbf{f}_{T,n}(t) \in \mathbb{R}^3$ is the tip force in $\mathcal{F}_n$. Define the relative displacement and velocity between the arm tip and beam tip as

$$\mathbf{d}_p(t) := \mathbf{p}_{c,n}(t) - \mathbf{p}_{T,n}(t), \qquad \mathbf{d}_v(t) := \mathbf{v}_{c,n}(t) - \mathbf{v}_{T,n}(t).$$

Let $\hat{\mathbf{a}}_n(t) := \mathbf{R}^n_b(t)\hat{\mathbf{a}}_b$ denote the contact-arm axis in $\mathcal{F}_n$. Define the scalar contact penetration and penetration rate as

$$\delta(t) := \max\{0,\ \hat{\mathbf{a}}_n(t)^{\top}\mathbf{d}_p(t)\},\ \dot{\delta}(t) := \mathbb{I}_{\{\delta(t)>0\}}\,\hat{\mathbf{a}}_n(t)^{\top}\mathbf{d}_v(t).$$

Here $\mathbb{I}_{\{\delta(t)>0\}} = 1$ in contact and 0 otherwise. With contact stiffness $k_c > 0$ and damping $b_c \ge 0$, the contact force is

$$\mathbf{f}_{T,n}(t) := \max\{0,\ k_c\,\delta(t) + b_c\,\dot{\delta}(t)\}\,\hat{\mathbf{a}}_n(t). \tag{9}$$

The external wrench on the vehicle in $\mathcal{F}_b$ is

$$\mathbf{w}_{e,b}(t) := \begin{bmatrix} -\mathbf{R}^b_n(t)\,\mathbf{f}_{T,n}(t) \\ \mathbf{r}_{c,b}(t) \times \big(-\mathbf{R}^b_n(t)\,\mathbf{f}_{T,n}(t)\big) \end{bmatrix}.$$

The prescribed tip offset $s(t)$ ramps to $s_\star$, holds for T_{hold}, and ramps down symmetrically. The initial ramp is

$$s(t) = \frac{s_\star}{2}\left(1 - \cos\left(\pi\frac{t - T_1}{T_{\mathrm{ramp}}}\right)\right), \quad t \in [T_1,\ T_1 + T_{\mathrm{ramp}}].$$

In all simulations, $0 \le s(t) \le 0.02$ m, $s_\star = 0.02$ m, $T_1 = 10$ s, $T_{\mathrm{ramp}} = 3$ s, and $T_{\mathrm{hold}} = 20$ s. The contact wrench is applied only for $t \ge T_1$, with $\mathbf{w}_{e,b}(t) = \mathbf{0}$ for $t < T_1$. Let $T_c := [t_c^-,\ t_c^+]$ denote the contact interval, with $t_c^- \ge T_1$ and $|T_c| := t_c^+ - t_c^-$. Two interaction cases are considered: y_b-axis and z_b-axis interaction with parametric uncertainty.

http://www.imavs.org/

Simulations are performed in MATLAB R2024b with Simscape Multibody using the vehicle parameters in Table 1 and a single controller parameter set across all cases with $T_s = 1$ ms. For each $q \in \{x_n, y_n, z_n, \psi\}$, the admittance is second order with $M_q = 1$, $\zeta_q = 1$, $T^{\mathrm{c}}_{\mathrm{set},q} = 2$ s, $\omega_{n,q} = 4/(\zeta_q T^{\mathrm{c}}_{\mathrm{set},q})$, $K_q = M_q\omega^2_{n,q}$, and $C_q = 2\zeta_q M_q \omega_{n,q}$. The admittance reference model uses $T_{\mathrm{set},q} = 1.5\,T^{\mathrm{c}}_{\mathrm{set},q}$ and $\zeta_q = 1$, with $\Gamma_{q1} = \Gamma_{q2} = \Gamma_{qr} = \Gamma_{qd} = 50$ and the adaptive admittance parameters initialized at the fixed-admittance values. The position MRAC reference model is critically damped with $T^{\mathrm{p}}_{\mathrm{set}} = 1$ s. The contact force limit $F_{\max,c}$ is set to 60% of the available thrust margin above hover and is enforced by saturating the penetration at $\delta_{\max}$ so that (9) does not exceed $F_{\max,c}$. All beam and contact parameters are listed in Table 2.

Table 1: Vehicle and rotor model parameters.

Parameter	Value
Vehicle mass (m)	0.95 kg
Thrust coefficient (k_f)	1.21×10^{-6} N s^2/rad^2
Reaction torque coefficient (k_m)	1.04×10^{-8} N m s^2/rad^2
Motor time constant (τ)	0.15 s
Min rotor speed ($\omega_{\min}$)	350 rad/s
Max rotor speed ($\omega_{\max}$)	1774 rad/s
Rotor tilt angle (β)	31°
Arm length (l)	88.5 mm
Top rotor vertical offset (h_t)	35 mm
Bottom rotor vertical offset (h_b)	20 mm

Table 2: Beam and contact parameters.

Parameter	Value
Beam length ($L_{\mathcal{B}}$)	1 m
Tip mass ($\mathbf{M}_{\mathcal{B}}$)	diag(0.10, 0.02, 0.02) kg
Tip damping ($\mathbf{C}_{\mathcal{B}}$)	diag(204.74, 0.25, 0.25) N s/m
Tip stiffness ($\mathbf{K}_{\mathcal{B}}$)	diag(2.6×10^6, 10.78, 10.78) N/m
Contact force limit ($F_{\max,c}$)	12 N
Penetration limit ($\delta_{\max}$)	0.02 m
Contact stiffness (k_c)	600 N/m
Contact damping (b_c)	2.41 N s/m

4.1 y_b-axis interaction with parametric uncertainty

For the y_b case, $\hat{\mathbf{a}}_b = [0 \;\; 1 \;\; 0]^\top$ and $\hat{\mathbf{a}}_n(t) = \mathbf{R}^n_b(t)\hat{\mathbf{a}}_b$, as shown in Fig. 3. The UAV reaches $\mathbf{p}_{T0,n}$ by T_{arr}, after which $\mathbf{w}_{e,b}(t)$ is applied. Results compare FA with constant parameters against adaptive admittance (AA) with online updates and MRAC augmentation. Fig. 4 shows AA position tracking over the full task, while FA–AA differences are evaluated in the contact-response plots. Fig. 5 compares $\hat{\mathbf{a}}_n^\top \mathbf{p}_n$ and $\hat{\mathbf{a}}_n^\top \mathbf{q}_{sp,n}$. AA has smaller post-contact tracking error and reduced oscillations, indicating better interaction-axis tracking. Figs. 6 and 7 show greater beam-tip deformation during the prescribed arm motion, while the contact force remains below $F_{\max,c}$.

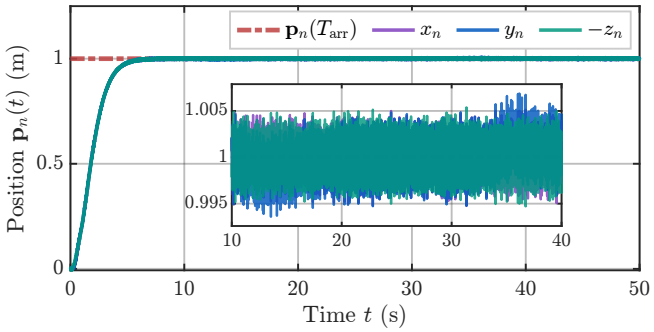


Figure 4: y_b-axis case: position components in $\mathcal{F}_n$.

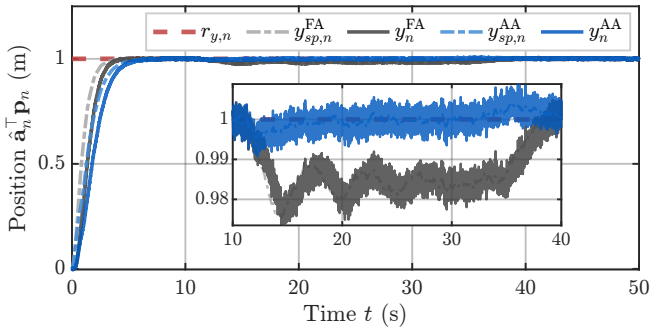
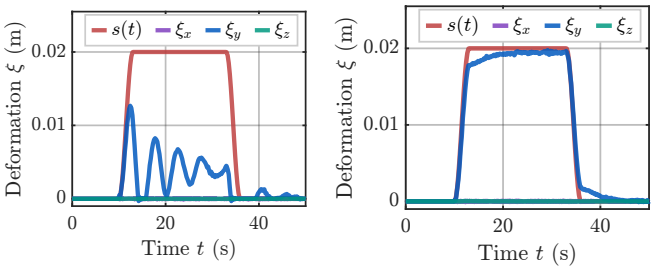


Figure 5: y_b-axis case: interaction-axis position and setpoint.



(a) Fixed Admittance. (b) Adaptive Admittance.

Figure 6: y_b-axis case: beam-tip deformation components.

For robustness, the beam matrices are scaled as $\mathbf{M}_{\mathcal{B}} \leftarrow \alpha\,\mathbf{M}_{\mathcal{B}}$, $\mathbf{C}_{\mathcal{B}} \leftarrow \alpha\,\mathbf{C}_{\mathcal{B}}$, and $\mathbf{K}_{\mathcal{B}} \leftarrow \alpha\,\mathbf{K}_{\mathcal{B}}$, with $\alpha \in \{0.25,\ 1.00,\ 1.25\}$. Over $T_c = [t_c^-, t_c^+]$, define $e_a(t) := \hat{\mathbf{a}}_n(t)^\top\big(\mathbf{p}_n(t) - \mathbf{q}_{sp,n}(t)\big)$ and $\mathrm{MAE}_a := \frac{1}{|T_c|}\int_{t_c^-}^{t_c^+} |e_a(t)|\,dt$. The force metrics are $F_{\hat{\mathbf{a}},\max} := \max_{t\in T_c} \big|\hat{\mathbf{a}}_n(t)^\top \mathbf{f}_{T,n}(t)\big|$ and $F_{\hat{\mathbf{a}},\mathrm{MA}} := \frac{1}{|T_c|}\int_{t_c^-}^{t_c^+} \big|\hat{\mathbf{a}}_n(t)^\top \mathbf{f}_{T,n}(t)\big|\,dt$, and the deformation metric is $\xi_{\max} := \max_{t\in T_c} \big|\hat{\mathbf{a}}_n(t)^\top \boldsymbol{\xi}(t)\big|$. Table 3 shows that AA reduces MAE_a and increases $\xi_{\max}$ for all α.

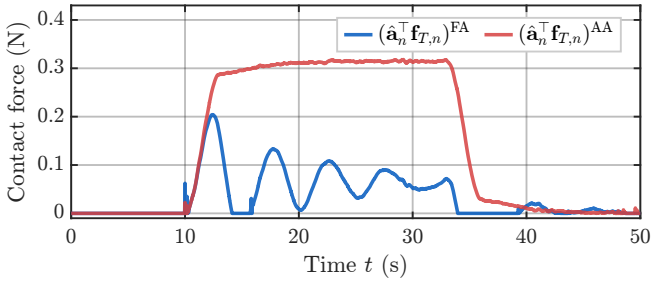


Figure 7: y_b-axis case: interaction-axis contact force.

Table 3: y_b-axis case: robustness under parameter scaling.

α	Metric	FA	AA
0.25	MAE_a (m)	6.46×10^{-4}	5.51×10^{-4}
	$F_{\hat{\mathbf{a}},\max}$ (N)	0.0718	0.0810
	$F_{\hat{\mathbf{a}},\mathrm{MA}}$ (N)	0.0267	0.0467
	$\xi_{\max}$ (m)	1.78×10^{-2}	2.01×10^{-2}
1.00	MAE_a (m)	1.50×10^{-3}	9.66×10^{-4}
	$F_{\hat{\mathbf{a}},\max}$ (N)	0.205	0.317
	$F_{\hat{\mathbf{a}},\mathrm{MA}}$ (N)	0.0449	0.183
	$\xi_{\max}$ (m)	1.27×10^{-2}	1.96×10^{-2}
1.25	MAE_a (m)	1.82×10^{-3}	1.13×10^{-3}
	$F_{\hat{\mathbf{a}},\max}$ (N)	0.237	0.394
	$F_{\hat{\mathbf{a}},\mathrm{MA}}$ (N)	0.0486	0.227
	$\xi_{\max}$ (m)	1.17×10^{-2}	1.95×10^{-2}

Greater deformation accompanies higher mean and peak contact forces, with $F_{\hat{\mathbf{a}},\max} \leq 0.394\ \mathrm{N} \ll F_{\max,c} = 12\ \mathrm{N}$.

4.2 z_b-axis interaction with parametric uncertainty

For the z_b case, $\hat{\mathbf{a}}_b = [0 \quad 0 \quad 1]^\top$, as shown in Fig. 8; FA and AA use the same contact profile. The AA full-task position response remains bounded and reaches the target, as in the y_b case. Figs. 9–11 show that AA reduces post-contact offset and oscillations and produces greater beam-tip deformation than FA under the same contact profile. The robustness study uses the same beam-parameter scaling as in Section 4.1, with $\alpha \in \{0.75,\ 1.00,\ 1.25\}$. Table 4 shows that AA reduces MAE_a and increases $\xi_{\max}$ for all α. Greater deformation accompanies higher mean and peak contact forces, with $F_{\hat{\mathbf{a}},\max} \leq 0.397\ \mathrm{N} \ll F_{\max,c} = 12\ \mathrm{N}$.

5 Conclusion

This paper presented an MRAC-based admittance architecture for aerial physical interaction. The admittance uses an MRAC input and Lyapunov-based parameter updates to follow a reference compliance model, while the MRAC position controller compensates matched translational uncertainty. The subsystem analyses establish bounded tracking and parameter errors under the stated assumptions. Simulations show reduced interaction-axis tracking error and greater beam-tip deformation relative to FA without exceeding the

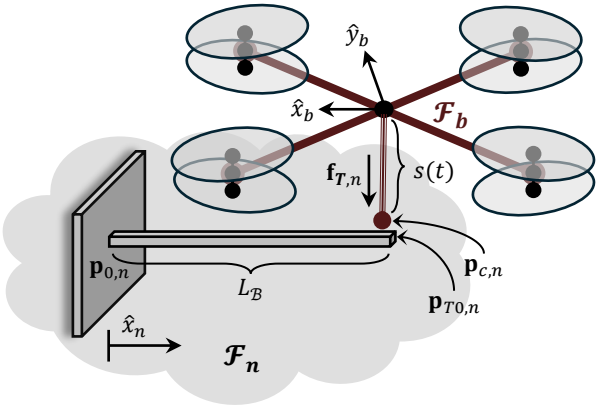


Figure 8: z_b-axis contact arm approaching a cantilever beam.

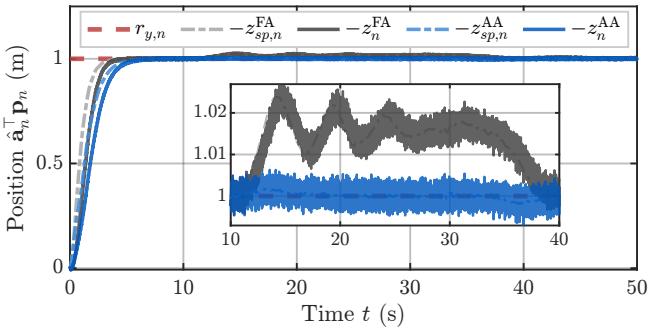
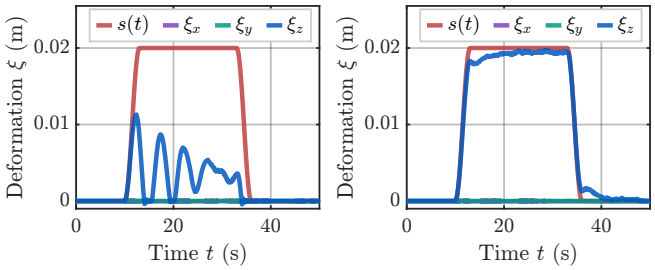


Figure 9: z_b-axis case: interaction-axis position and setpoint.



(a) Fixed Admittance. (b) Adaptive Admittance.

Figure 10: z_b-axis case: beam-tip deformation components.

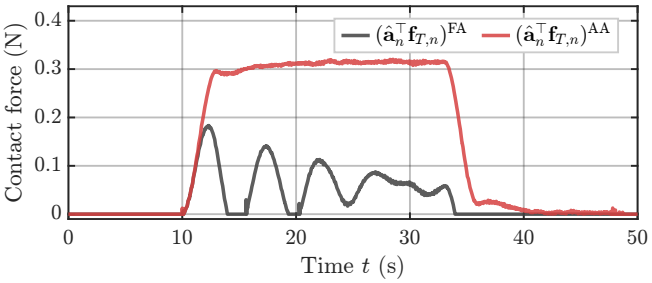


Figure 11: z_b-axis case: interaction-axis contact force.

http://www.imavs.org/

Table 4: z_b-axis case: robustness under parameter scaling.

α	Metric	FA	AA
0.75	MAE_a (m)	1.26×10^{-3}	1.25×10^{-4}
	$F_{\hat{\mathbf{a}},\max}$ (N)	0.152	0.242
	$F_{\hat{\mathbf{a}},\mathrm{MA}}$ (N)	0.0596	0.142
	$\xi_{\max}$ (m)	1.25×10^{-2}	1.99×10^{-2}
1.00	MAE_a (m)	1.71×10^{-3}	1.34×10^{-4}
	$F_{\hat{\mathbf{a}},\max}$ (N)	0.184	0.322
	$F_{\hat{\mathbf{a}},\mathrm{MA}}$ (N)	0.0654	0.189
	$\xi_{\max}$ (m)	1.13×10^{-2}	1.97×10^{-2}
1.25	MAE_a (m)	2.29×10^{-3}	1.43×10^{-4}
	$F_{\hat{\mathbf{a}},\max}$ (N)	0.210	0.397
	$F_{\hat{\mathbf{a}},\mathrm{MA}}$ (N)	0.0697	0.233
	$\xi_{\max}$ (m)	1.04×10^{-2}	1.96×10^{-2}

contact force limit. Future work will extend the Lyapunov analysis to the coupled adaptive loops and validate the proposed architecture on the FT-X8 hardware.

ACKNOWLEDGEMENTS

The research 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] K. Bodie, M. Brunner, M. Pantic, S. Walser, P. Pfändler, U. Angst, R. Siegwart, and J. Nieto. Active interaction force control for contact-based inspection with a fully actuated aerial vehicle. *IEEE Trans. Robot.*, 37(3):709–722, 2021.

[2] A. Alharbat, C. Gabellieri, A. Y. Mersha, and A. Franchi. Predictive admittance control for aerial physical interaction. *IEEE Robot. Autom. Lett.*, 10(11):11235–11242, 2025.

[3] M. Ryll, G. Muscio, F. Pierri, E. Cataldi, G. Antonelli, F. Caccavale, D. Bicego, and A. Franchi. 6D interaction control with aerial robots: The flying end-effector paradigm. *Int. J. Robot. Res.*, 38(9):1045–1062, 2019.

[4] H. N. Naser, H. A. Hashim, and M. Ahmadi. Human physical interaction based on UAV cooperative payload transportation system using adaptive backstepping and FNTSMC. In *Proc. Amer. Control Conf. (ACC)*, pages 450–456, 2025.

[5] S. Hamaza, I. Georgilas, and T. Richardson. Towards an adaptive-compliance aerial manipulator for contact-based interaction. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 7448–7453, 2018.

[6] L. Marković, M. Car, M. Orsag, and S. Bogdan. Adaptive stiffness estimation impedance control for achieving sustained contact in aerial manipulation. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 117–123, 2021.

[7] M. Orsag, C. Korpela, S. Bogdan, and P. Oh. Lyapunov based model reference adaptive control for aerial manipulation. In *Proc. Int. Conf. Unmanned Aircraft Syst.*, pages 966–973, 2013.

[8] M. Orsag, C. M. Korpela, S. Bogdan, and P. Y. Oh. Hybrid adaptive control for aerial manipulation. *J. Intell. Robot. Syst.*, 73(1):693–707, 2014.

[9] V. Sumathy and D. Ghose. Adaptive augmented torque control of a quadcopter with an aerial manipulator. *Drone Syst. Appl.*, 10(1):26–50, 2022.

[10] Z. A. Li and X. Li. Modeling and controlling of quadrotor aerial vehicle equipped with a gripper. *Measurement and Control*, 52(5-6):577–587, 2019.

[11] G. Baraban, M. Sheckells, S. Kim, and M. Kobilarov. Adaptive parameter estimation for aerial manipulation. In *Proc. Amer. Control Conf.*, pages 614–619, 2020.

[12] T. Maki, M. Zhao, F. Shi, K. Okada, and M. Inaba. Model reference adaptive control of multirotor for missions with dynamic change of payloads during flight. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 7433–7439, 2020.

[13] R. Rashad, J. B. C. Engelen, and S. Stramigioli. Energy tank-based wrench/impedance control of a fully actuated hexarotor: A geometric port-hamiltonian approach. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 6418–6424, 2019.

[14] P. H. M. Souza and K. Stol. Constrained dynamics of an aerial manipulator interacting with flexible cantilever beams. *IEEE/ASME Trans. Mechatronics*, 28(2):967–975, 2023.

[15] L. Meier, D. Honegger, and M. Pollefeys. PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 6235–6240, 2015.

[16] T. Tomić, C. Ott, and S. Haddadin. External wrench estimation, collision detection, and reflex reaction for flying robots. *IEEE Trans. Robot.*, 33(6):1467–1482, 2017.

[17] P. A. Ioannou and J. Sun. *Robust Adaptive Control*. Dover, Mineola, NY, 2012.

http://www.imavs.org/

Variable Admittance Control for Obstacle Separation in Aerial Manipulation

Pedro H. M. Souza, Karl Stol*, and Ishaq Hafez
Department of Mechanical and Mechatronics Engineering, University of Auckland, Auckland 1010, New Zealand

http://www.imavs.org/

Abstract

This paper presents variable admittance control for elastic obstacle separation using a fully actuated fixed-tilt octorotor aerial manipulator. The vehicle carries a single-actuator spreading mechanism that applies opposing contact forces to separate elastic obstacles while limiting the resultant interaction force on the vehicle. The controller modifies the reference motion using the estimated translational force and yaw moment, while the wrench estimator uses online recursive least-squares (RLS) correction to compensate for unmodeled aerodynamic effects. The admittance parameters vary channel-wise as functions of the estimated wrench component, obstacle distance, and actuator saturation, producing stiff free-flight tracking and compliant contact response without mode switching. An experiment demonstrates bounded vehicle motion during sustained separation of two elastic bars.

1 Introduction

Aerial manipulation enables unmanned aerial vehicles (UAVs) to perform tasks requiring physical interaction with the environment. Contact inspection, repair, human-UAV interaction, and plant sampling require the vehicle to regulate interaction forces while maintaining stable flight [1, 2, 3, 4]. In agricultural and ecological applications, the target region may be obstructed by branches, twigs, or other elastic obstacles [5, 6]. The vehicle may need to displace these obstacles before inspection or sampling, requiring force regulation and sufficient actuation authority.

Fully actuated multirotors are suitable for aerial physical interaction because they reduce the coupling between translational force generation and attitude regulation. Rashad et al. [7] reviewed fully actuated multirotor concepts, while Ryll et al. [8] and Bodie et al. [9] demonstrated interaction control on platforms capable of regulating interaction forces. Existing designs use active rotor tilting, thrust-vectoring mechanisms, or fixed-tilt rotor layouts [10, 11, 7]. Active rotor-tilting and thrust-vectoring designs increase body-wrench authority, but require additional actuators and more complex control allocation. Fixed-tilt platforms provide a simpler alternative by using canted rotors to generate horizontal thrust components without requiring large roll or pitch angles [12, 13]. A prior flexible-rod contact experiment with the smaller fixed-tilt platform is shown in Fig. 1. However, this simpler configuration comes at the cost of reduced endurance. When contact loads are transmitted through the vehicle, the achievable contact force is bounded by the available horizontal thrust. Elastic obstacle separation, such as separating flexible tree branches, can require large local contact forces while limiting the resultant force transmitted to the UAV. This motivates a spreading mechanism that generates the separation load through opposing contact forces, reducing the horizontal thrust required from the UAV.

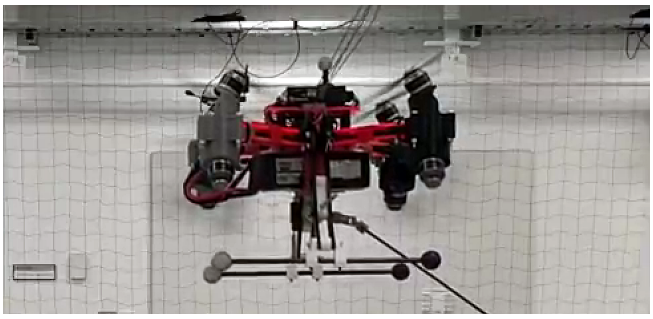
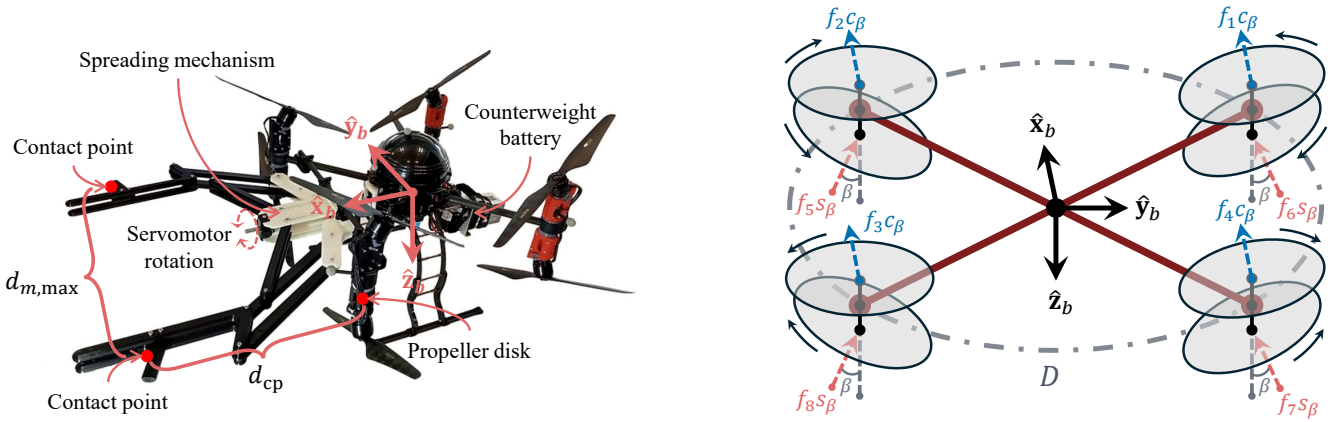


Figure 1: Contact experiment between the smaller fixed-tilt multirotor and a vertical flexible rod [13].

The mechanism increases local contact-force capability, but interaction control remains necessary. Admittance control addresses this problem by mapping the estimated external wrench to reference motion through virtual inertia, damping, and stiffness [4, 14, 15]. In free flight, high virtual stiffness improves reference tracking and disturbance rejection. During contact, excessive virtual stiffness can increase interaction forces and destabilize the coupled vehicle–environment response, while excessive compliance can degrade positioning accuracy. A fixed-admittance controller uses constant virtual parameters in free flight and contact, requiring a compromise between tracking and compliance. Variable admittance instead updates the virtual inertia, damping, and stiffness online, producing stiff free-flight behavior and compliant contact response without mode switching [9, 16, 17].

This paper contributes a task-specific system that integrates variable-admittance control and wrench estimation with an opposing-force spreading mechanism for elastic obstacle separation. The single-actuator mechanism generates

*Email address: k.stol@auckland.ac.nz



(a) Spreading mechanism geometry.

(b) Rotor geometry.

Figure 2: FT-X8 geometry: (a) spreading direction $\hat{\mathbf{s}}_b = \hat{\mathbf{y}}_b$ and (b) $c_\beta = \cos\beta$, $s_\beta = \sin\beta$.

opposing local contact forces, while four channel-wise admittance filters vary their virtual inertia and stiffness according to the corresponding estimated wrench component, obstacle distance, and actuator saturation. Online recursive least-squares (RLS) estimation corrects the applied rotor-wrench and gravity-induced moment terms used by the wrench estimator, and the resulting external-wrench estimate drives the translational and yaw admittance channels. The resulting system operates continuously between free flight and contact without explicit mode switching and is evaluated during sustained bilateral spreading.

2 System Overview

The aerial manipulator is a fully actuated fixed-tilt octorotor, denoted FT-X8. The FT-X8 extends the compact fixed-tilt platform used for flexible-cantilever interaction in [12, 13] by incorporating an opposing-force spreading mechanism. Its eight rotors are arranged in four coaxial pairs with a common cant angle β, as shown in Fig. 2. This layout generates body-frame horizontal thrust components and enables lateral translation while maintaining a near-level attitude. The rotor-to-rotor diameter D is approximately 37% larger than that of the platform in [13]. The rotors regulate vehicle motion, while the spreading mechanism supplies the local contact forces required for elastic obstacle separation.

Let $\mathcal{F}_n$ denote the inertial frame and $\mathcal{F}_b$ the body-fixed frame attached to the vehicle center of mass. The inertial frame follows the North-East-Down (NED) convention, with $\hat{\mathbf{z}}_n = [\,0 \;\; 0 \;\; 1\,]^\top$ pointing downward. The body-frame unit vectors $\hat{\mathbf{x}}_b$, $\hat{\mathbf{y}}_b$, and $\hat{\mathbf{z}}_b$ are shown in Fig. 2. The vehicle position and attitude are $\mathbf{p}_n(t) := [\,x_n(t) \;\; y_n(t) \;\; z_n(t)\,]^\top$ and $\boldsymbol{\eta}(t) := [\,\phi(t) \;\; \theta(t) \;\; \psi(t)\,]^\top$, where $\boldsymbol{\eta}$ follows the ZYX Euler convention. The rotation matrix $\mathbf{R}_b^n(t) \in SO(3)$ maps vectors from $\mathcal{F}_b$ to $\mathcal{F}_n$, with $\mathbf{R}_n^b(t) := (\mathbf{R}_b^n(t))^\top$. The body-frame linear and angular velocities are $\mathbf{v}_b(t) := [\,v_{x,b}(t) \;\; v_{y,b}(t) \;\; v_{z,b}(t)\,]^\top$ and $\boldsymbol{\omega}_b(t) := [\,p(t) \;\; q(t) \;\; r(t)\,]^\top$. The external wrench is $\mathbf{w}_{e,b}(t) := [\,\mathbf{f}_{e,b}^\top(t) \;\; \boldsymbol{\tau}_{e,b}^\top(t)\,]^\top$.

The rigid-body dynamics and fixed-tilt rotor model follow [12, 13]. Let $m > 0$, $g > 0$, and $\mathbf{J} \succ 0$ denote the total mass, gravitational acceleration, and inertia matrix. The rotor-generated body wrench is $\mathbf{w}_b(t) := [\,\mathbf{f}_b^\top(t) \;\; \mathbf{m}_b^\top(t)\,]^\top$. With the rotor thrust vector $\mathbf{f}_r(t) := [\,f_1(t) \;\cdots\; f_8(t)\,]^\top$, the allocation model is $\mathbf{w}_b(t) = \boldsymbol{\Lambda}\mathbf{f}_r(t)$, where $\boldsymbol{\Lambda} \in \mathbb{R}^{6\times 8}$. Since $\mathrm{rank}(\boldsymbol{\Lambda}) = 6$, the vehicle is fully actuated.

The spreading mechanism is mounted on the front arms and driven by a Dynamixel MX-64 servomotor through a lead-screw linkage. It produces approximately parallel motion at the two contact points during actuation. As shown in Fig. 2a, the maximum contact-point separation is denoted $d_{m,\max}$, and the contact-point clearance d_{cp} is the minimum distance from either contact point to the nearest propeller disk. The 6S LiPo battery counterweights the manipulator to reduce the center-of-mass offset.

Let $\hat{\mathbf{s}}_b := \hat{\mathbf{y}}_b$ denote the body-frame spreading direction. For the two obstacle contacts, the contact forces are

$$\mathbf{f}_{c,1,b} = F_1\hat{\mathbf{s}}_b, \qquad \mathbf{f}_{c,2,b} = -F_2\hat{\mathbf{s}}_b,$$

where $F_1, F_2 \geq 0$ are the contact force magnitudes. The reaction force transmitted to the vehicle is

$$\mathbf{f}_{m,b} = -\mathbf{f}_{c,1,b} - \mathbf{f}_{c,2,b} = (F_2 - F_1)\hat{\mathbf{s}}_b.$$

For symmetric obstacle separation, the bars have approximately equal stiffness, so $F_1 \approx F_2$. The mechanism therefore applies a pair of opposing local contact forces, producing a small resultant force on the vehicle. The variable-admittance channels use an estimate of the resultant external wrench, $\hat{\mathbf{w}}_{e,b}$, rather than the individual contact forces F_1 and F_2, because only their net effect enters the vehicle dynamics. For unequal bar stiffness, $F_1 \neq F_2$, so $\mathbf{f}_{m,b} \neq \mathbf{0}$. Because the contact points are offset from the vehicle center of mass, this imbalance may also produce an external moment about

http://www.imavs.org/

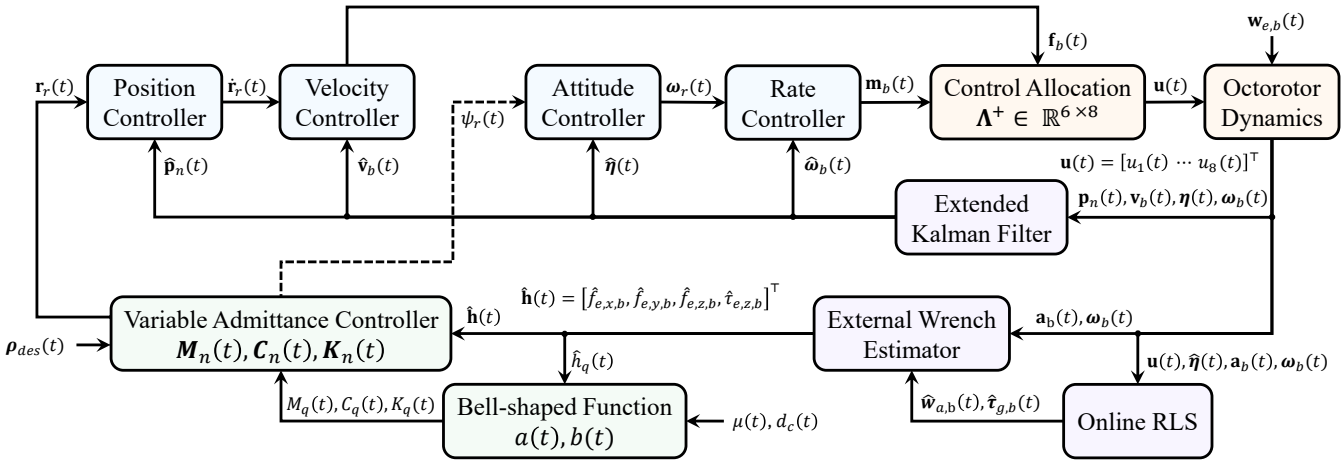


Figure 3: Variable admittance control architecture.

Table 1: FT-X8 with spreading mechanism parameters.

Parameter	Value
Motor model	T-Motor U5-400
Propeller	CF NS16x5.4
Rotor cant angle, β	30°
Rotor-to-rotor diameter, D	0.625 m
Total mass, m	7.1 kg
Manipulator mass	1.8 kg
Maximum horizontal thrust	19.7 N
Manipulator maximum load	115 N
Maximum separation, $d_{m,\max}$	0.463 m
Contact-point clearance, d_{cp}	0.225 m

$\hat{\mathbf{z}}_b$. The estimated force and yaw moment drive the translational and yaw admittance channels, respectively. Performance under this imbalance depends on wrench-estimation accuracy and available body-wrench authority. The attainable local contact force is constrained primarily by the manipulator maximum load, whereas the available horizontal thrust constrains the resultant force transmitted to the vehicle.

The vehicle pose is measured by a Vicon motion-capture system. An offboard MATLAB/Simulink interface exchanges position setpoints and pose measurements with a Pixhawk 4 running the PX4 flight stack [18] over MAVLink. The spreading mechanism is commanded separately, while PX4 tracks the admittance-shaped reference.

3 VARIABLE ADMITTANCE CONTROL

The outer admittance loop modifies the desired reference to generate an admittance-shaped reference, which is tracked by the inner PX4 flight controller, as shown in Fig. 3. The desired translational reference is expressed in $\mathcal{F}_b$ as

$$\mathbf{r}_{\mathrm{des}}(t) := [\, x_{\mathrm{des}}(t) \;\; y_{\mathrm{des}}(t) \;\; z_{\mathrm{des}}(t) \,]^\top,$$

with desired yaw angle $\psi_{\mathrm{des}}(t)$. The admittance loop generates the compliant translational reference

$$\mathbf{r}_r(t) := [\, x_r(t) \;\; y_r(t) \;\; z_r(t) \,]^\top$$

and yaw reference $\psi_r(t)$. The combined references are

$$\boldsymbol{\rho}_{\mathrm{des}}(t) := [\, \mathbf{r}_{\mathrm{des}}^\top(t) \;\; \psi_{\mathrm{des}}(t) \,]^\top, \quad \boldsymbol{\rho}_r(t) := [\, \mathbf{r}_r^\top(t) \;\; \psi_r(t) \,]^\top.$$

The admittance loop uses the body-frame external wrench estimate $\hat{\mathbf{w}}_{e,b}(t) := [\, \hat{\mathbf{f}}_{e,b}^\top(t) \;\; \hat{\boldsymbol{\tau}}_{e,b}^\top(t) \,]^\top$. This estimate is obtained by combining an acceleration-based force observer and an angular-momentum-based external-moment observer using accelerometer and gyroscope measurements and the rotor-generated wrench, without requiring a force/torque sensor. The PX4 state estimator provides the vehicle-state inputs to these observers. The force and moment observers compute $\hat{\mathbf{w}}_{e,b}(t)$ using RLS-corrected model terms. The specific force measured by the accelerometer, expressed in $\mathcal{F}_b$, is

$$\mathbf{a}_b(t) = \mathbf{R}_n^b(t)\big(\ddot{\mathbf{p}}_n(t) - g\hat{\mathbf{z}}_n\big).$$

Let $\hat{\mathbf{w}}_{a,b}(t) := [\, \hat{\mathbf{f}}_{a,b}^\top(t) \;\; \hat{\boldsymbol{\tau}}_{a,b}^\top(t) \,]^\top$ denote the RLS-corrected applied rotor wrench, and let $\hat{\boldsymbol{\tau}}_{g,b}(t)$ denote the moment from the estimated center-of-mass offset. The body-frame external force and moment observers are

$$\hat{\mathbf{f}}_{e,b}(t) = \mathbf{K}_f \int_0^t \Big(m\mathbf{a}_b(\sigma) - \hat{\mathbf{f}}_{a,b}(\sigma) - \hat{\mathbf{f}}_{e,b}(\sigma) \Big) d\sigma, \quad (1)$$

$$\hat{\boldsymbol{\tau}}_{e,b}(t) = \mathbf{K}_\tau \Bigg[\mathbf{J}\boldsymbol{\omega}_b(t) - \int_0^t \Big(\mathbf{J}\boldsymbol{\omega}_b(\sigma) \times \boldsymbol{\omega}_b(\sigma) + \hat{\boldsymbol{\tau}}_{g,b}(\sigma) + \hat{\boldsymbol{\tau}}_{a,b}(\sigma) - \hat{\boldsymbol{\tau}}_{e,b}(\sigma) \Big) d\sigma \Bigg]. \quad (2)$$

Here, $\mathbf{K}_f \succ 0$ and $\mathbf{K}_\tau \succ 0$ are force and moment estimator gain matrices. A small dead zone is applied to the estimated

http://www.imavs.org/

http://www.imavs.org/

wrench components to suppress sensor-noise-induced residuals near the zero-wrench condition.

The RLS block updates estimates of the constant parameters used in $\hat{\mathbf{w}}_{a,b}(t)$ and $\hat{\boldsymbol{\tau}}_{g,b}(t)$ [19]. It corrects the rotor-wrench model for aerodynamic interference in the canted coaxial pairs and the center-of-mass offset introduced by the spreading mechanism. The RLS update corrects only the rotor-wrench and gravity-induced moment terms online. The estimator is initialized from free-flight data and uses actuator commands and onboard inertial measurements. The corrected terms are used in the wrench estimator before the residual external wrench is computed. The RLS block does not estimate contact forces directly or change the admittance law.

For the admittance law, the full wrench estimate is reduced to the translational force components and the body z-axis moment. The four-channel admittance input is

$$\hat{\mathbf{h}}(t) := [\, \hat{f}_{e,x,b}(t) \;\; \hat{f}_{e,y,b}(t) \;\; \hat{f}_{e,z,b}(t) \;\; \hat{\tau}_{e,z,b}(t) \,]^\top,$$

where $\hat{\tau}_{e,z,b}(t)$ is the estimated moment about $\hat{\mathbf{z}}_b$. Roll and pitch remain regulated by the inner attitude controller and are therefore excluded from the admittance input.

The virtual inertia, damping, and stiffness matrices are

$$\mathbf{M}_n(t) = \operatorname{diag}\big(M_x(t), M_y(t), M_z(t), M_\psi(t)\big),$$
$$\mathbf{C}_n(t) = \operatorname{diag}\big(C_x(t), C_y(t), C_z(t), C_\psi(t)\big),$$
$$\mathbf{K}_n(t) = \operatorname{diag}\big(K_x(t), K_y(t), K_z(t), K_\psi(t)\big).$$

The diagonal structure gives four decoupled admittance filters for the translational components in $\mathcal{F}_b$ and yaw. For each channel $q \in \mathcal{Q} := \{x, y, z, \psi\}$, the scalar admittance input is

$$\hat{h}_q(t) := \begin{cases} \hat{f}_{e,q,b}(t), & q \in \{x, y, z\}, \\ \hat{\tau}_{e,z,b}(t), & q = \psi. \end{cases}$$

The channel-wise admittance equation for q is

$$M_q(t)\big(\ddot{\rho}_{r,q}(t) - \ddot{\rho}_{\mathrm{des},q}(t)\big) + C_q(t)\big(\dot{\rho}_{r,q}(t) - \dot{\rho}_{\mathrm{des},q}(t)\big) + K_q(t)\big(\rho_{r,q}(t) - \rho_{\mathrm{des},q}(t)\big) = \hat{h}_q(t).$$

Each bell-shaped law uses the scalar channel input $\hat{h}_q(t)$, not the full vector $\hat{\mathbf{h}}(t)$. The inertia and stiffness are defined by generalized bell-shaped functions [20]. For each $q \in \mathcal{Q}$,

$$M_q(t) = M_{q,\min} + \frac{M_{q,\max} - M_{q,\min}}{1 + \left|\frac{\hat{h}_q(t)}{a(t)}\right|^{2b(t)}}, \tag{3}$$

and

$$K_q(t) = K_{q,\min} + \frac{K_{q,\max} - K_{q,\min}}{1 + \left|\frac{\hat{h}_q(t)}{a(t)}\right|^{2b(t)}}. \tag{4}$$

The bounds satisfy $0 < M_{q,\min} < M_{q,\max}$ and $0 < K_{q,\min} < K_{q,\max}$. For small wrench components, (3) and (4) remain near $M_{q,\max}$ and $K_{q,\max}$, producing stiff tracking and disturbance rejection. For larger wrench components, $M_q(t)$ and $K_q(t)$ approach $M_{q,\min}$ and $K_{q,\min}$, increasing compliance during contact. The damping coefficient is set to

$$C_q(t) = 2\sqrt{M_q(t) K_q(t)}, \qquad q \in \mathcal{Q},$$

which sets each channel damping ratio to one.

The distance to the obstacles and actuator saturation modify the bell-shaped admittance law. Let $d_c(t) \geq 0$ denote the minimum distance from the manipulator contact points to the two elastic obstacles, with $d_c(t) = 0$ during contact. The distance-dependent shape parameter is

$$b(t) = b_1 - b_2 \exp\left(-\frac{4 d_c(t)}{b_3}\right), \tag{5}$$

where $b_1 > b_2 > 0$ and $b_3 > 0$. When the vehicle is far from the obstacles, $d_c(t)$ is large and $b(t) \approx b_1$, producing a sharper transition that keeps the virtual inertia and stiffness near their maximum values for small estimated wrench components. As the vehicle approaches contact, $b(t)$ decreases toward $b_1 - b_2$, and the smoother transition makes the virtual inertia and stiffness decrease at lower wrench magnitudes.

The bell-shaped law is also adjusted by actuator saturation so that compliance increases near actuator limits. Let $\mathbf{u}(t) := [\, u_1(t) \;\cdots\; u_8(t) \,]^\top \in [u_{\min}, u_{\max}]^8$ denote the motor pulse-width-modulation (PWM) command vector, where $u_{\min} = 1000\ \mu$s, $u_{\max} = 2000\ \mu$s, and $u_0 := (u_{\min} + u_{\max})/2 = 1500\ \mu$s. The raw saturation factor is

$$\mu_r(t) = \frac{\|\mathbf{u}(t) - u_0 \mathbf{1}_8\|_\infty}{u_{\max} - u_0}, \tag{6}$$

so that $\mu_r(t) = 0$ when all commands are centered at $1500\ \mu$s and $\mu_r(t) = 1$ when at least one motor reaches a saturation limit. A first-order low-pass filter is applied to $\mu_r(t)$ to obtain the filtered saturation factor $\mu(t) \in [0, 1]$ and avoid rapid parameter variation. The bell-width parameter is

$$a(t) = a_1 - \mu(t), \tag{7}$$

where $a_1 > 1$ sets the maximum bell width. As actuator saturation is approached, $\mu(t)$ increases and $a(t)$ decreases, so $M_q(t)$ and $K_q(t)$ decrease at smaller wrench magnitudes, increasing compliance near actuator limits.

Equations (3)–(7) define the admittance parameters as functions of the estimated wrench component $\hat{h}_q(t)$, obstacle distance $d_c(t)$, and filtered saturation factor $\mu(t)$. Thus, the same admittance law applies in free flight and contact, without explicit switching between modes.

Table 2 lists the admittance bounds for the translation axes (x_b, y_b, z_b) and yaw ψ, the bell-shaped law parameters, and the saturation-filter time constant. The bounds were selected so that $M_q(t)$ and $K_q(t)$ remain high in free flight and decrease during contact and near actuator saturation.

Table 2: Admittance parameters for obstacle separation.

Parameter	Translation axes	Yaw
$M_{\min}$	1 kg	10 kg m^2
$M_{\max}$	10 kg	100 kg m^2
$K_{\min}$	25 N/m	8 N m/rad
$K_{\max}$	50 N/m	10 N m/rad
Bell-shaped law parameters $a_1 = 2.5$, $b_1 = 5$, $b_2 = 3.5$, $b_3 = 2$ m		
Saturation-filter time constant $\tau_\mu = 5$ s		

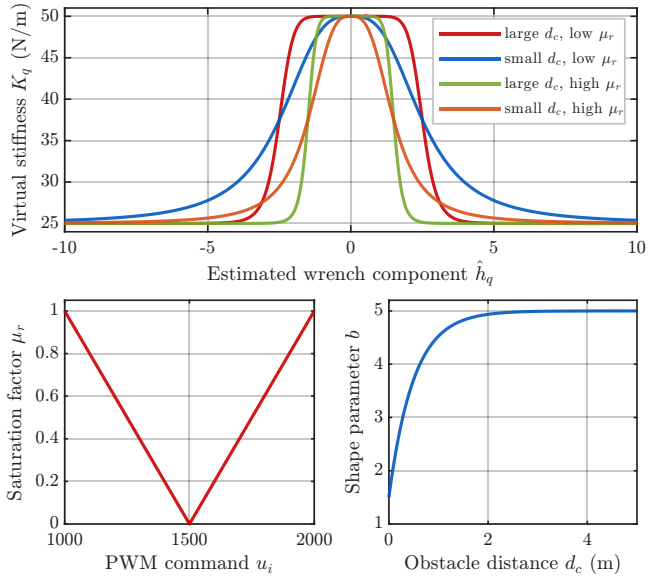


Figure 4: Effect of $a(t)$ and $b(t)$ on the stiffness law.

Fig. 4 illustrates the stiffness variation in (4) using the translational stiffness bounds from Table 2. The four curves correspond to combinations of obstacle distance and actuator saturation factor. For large d_c, $b \approx b_1$, while near contact $b \approx b_1 - b_2$. Low saturation gives $\mu \approx 0$ and $a \approx a_1$, while high saturation increases μ and reduces a. A larger a keeps K_q near $K_{\max}$ over a wider range of $|\hat{h}_q|$, so the drop toward $K_{\min}$ occurs at larger wrench magnitudes. A larger b keeps the response stiff for small wrench magnitudes and makes the transition to the compliant region sharper. The inertia law in (3) follows the same dependence on $\hat{h}_q(t)$, $a(t)$, and $b(t)$.

4 EXPERIMENTAL VALIDATION

The variable admittance controller is evaluated on the FT-X8 aerial manipulator during elastic obstacle separation. The experiment uses the equal-stiffness test rig shown in Fig. 5, where two vertical bars are restrained by elastic bands to emulate elastic obstacles. As the spreading mechanism opens, it contacts and separates the two bars while the vehicle remains under position control. This setup evaluates whether the controller maintains bounded vehicle motion while the mechanism applies large local contact forces during sustained bilateral contact.

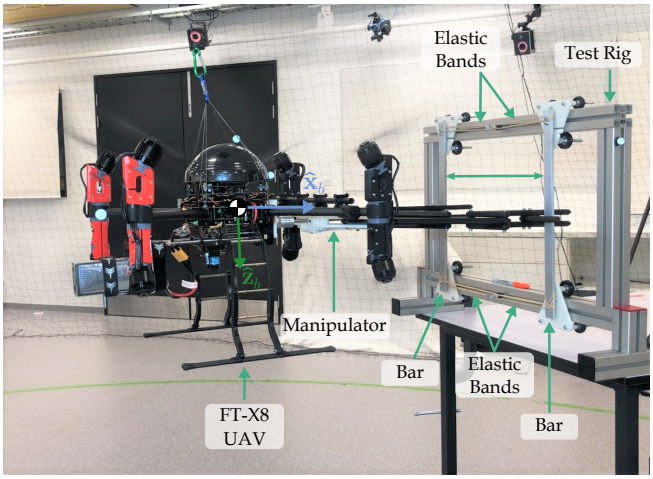


Figure 5: Experimental setup for elastic obstacle separation.

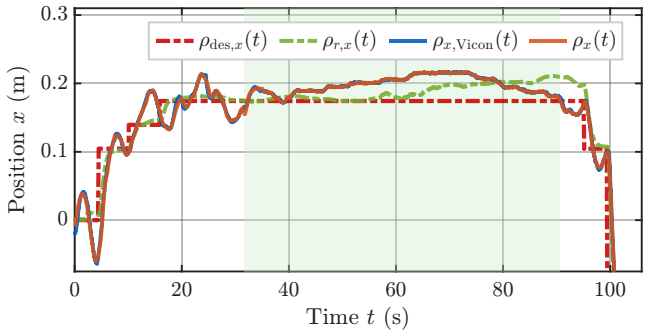
The vehicle is commanded to hover with the admittance controller active and then approach the center of the test rig until the spreading mechanism is positioned between the bars and the vehicle body-frame $\hat{\mathbf{y}}_b$ axis is aligned with the test-rig normal. After the mechanism is inserted between the bars, the servomotor drives the spreading mechanism gradually through its opening and closing motion, with both contact points remaining in contact with the elastic bars. The measured stiffness of each bar along $\hat{\mathbf{y}}_b$ is 387 N/m. Based on this stiffness and the full-opening displacement, full opening corresponds to an inferred local force of nearly 88 N per bar. This inferred local force is 4.5 times the maximum horizontal thrust listed in Table 1.

In Figs. 6 and 7, the green shaded region denotes the bilateral spreading phase, during which both contact points remain in contact with the bars as the mechanism opens and closes. The unshaded portions correspond to vehicle positioning before and after bilateral contact. In Fig. 6, for each plotted component $q \in \{x, y, z, \psi\}$, the desired reference is denoted by $\rho_{\mathrm{des},q}$, the admittance-shaped reference by $\rho_{r,q}$, the PX4 estimate by ρ_q, and the Vicon measurement by $\rho_{q,\mathrm{Vicon}}$.

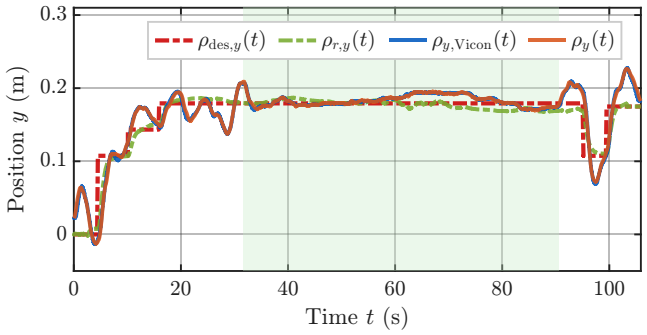
Fig. 6 shows the vehicle motion during spreading. The PX4 estimate and Vicon measurement follow the same horizontal-motion trend. During bilateral spreading, the admittance-shaped reference intentionally deviates from the desired reference because the variable admittance law modifies the reference in response to the estimated interaction wrench instead of enforcing rigid tracking of the desired reference. The horizontal reference deviations persist throughout the bilateral phase, indicating sustained compliance rather than a short contact transient. The desired vertical reference remains nearly constant, while the Vicon measurement varies by 0.041 m peak to peak over the interval shown.

http://www.imavs.org/

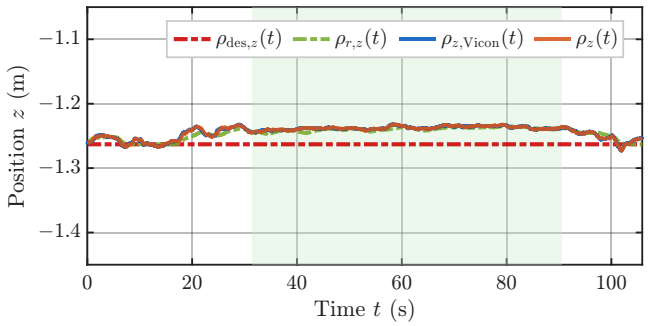
http://www.imavs.org/



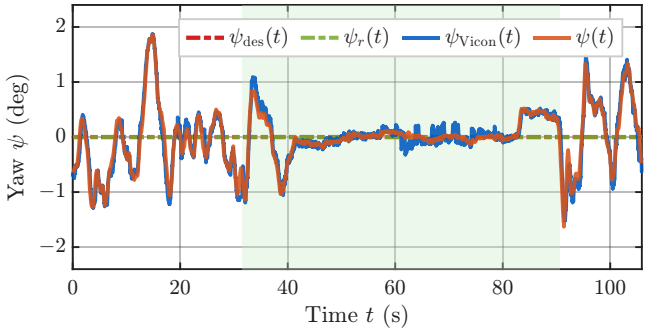
(a) x response.



(b) y response.

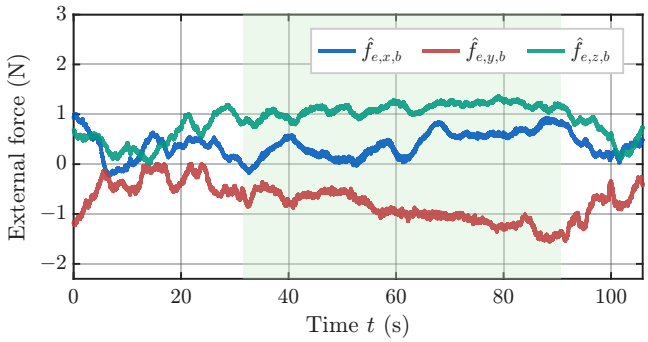


(c) z response.

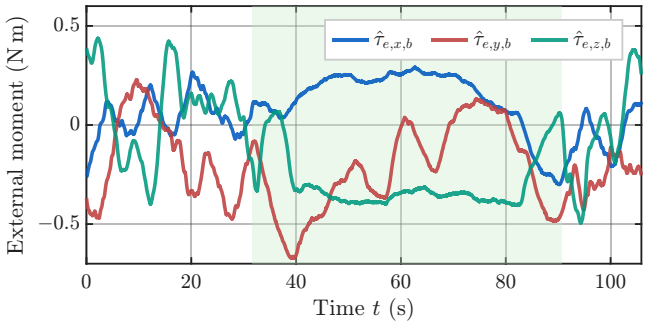


(d) Yaw response.

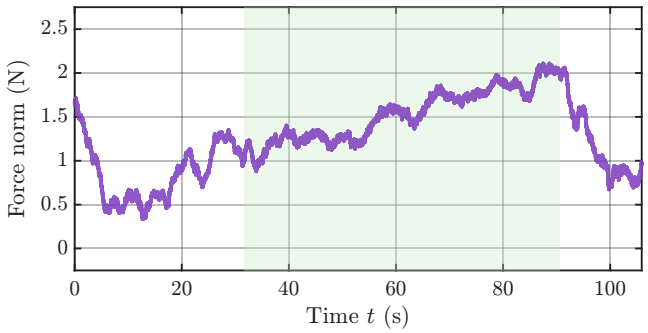
Figure 6: Vehicle motion during the spreading experiment. The shaded region marks the bilateral spreading phase.



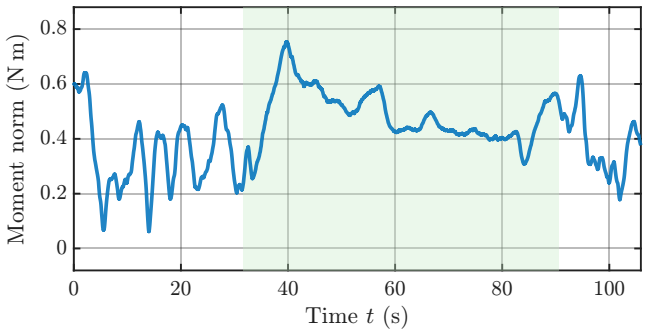
(a) External force components $\hat{f}_{e,q,b}$, $q \in \{x, y, z\}$.



(b) External moment components $\hat{\tau}_{e,q,b}$, $q \in \{x, y, z\}$.



(c) External force norm $\|\hat{\mathbf{f}}_{e,b}\|$.



(d) External moment norm $\|\hat{\boldsymbol{\tau}}_{e,b}\|$.

Figure 7: RLS-corrected external wrench estimate during the spreading experiment, expressed in the body frame. The component plots show the dominant interaction directions, while the norms summarize the force and moment magnitudes.

http://www.imavs.org/

The root-mean-square (RMS) yaw-tracking error relative to ψ_r over the bilateral spreading phase was 0.293° for the PX4 estimate ψ and 0.310° for the Vicon measurement ψ_{Vicon}. The corresponding maximum absolute yaw-tracking errors were 1.14° and 1.19°, respectively. During bilateral spreading, the maximum raw saturation factor computed from the eight motor PWM commands using (6) was $\mu_{r,\max} = 0.334$, corresponding to a minimum PWM margin of $\Delta u_{\min} = 333$ μs to the nearest saturation limit. The motor PWM commands remained within 1422–1667 μs during this phase, and no motor command reached the 1000 or 2000 μs limits. The response therefore shows bounded vehicle motion while retaining actuator authority during separation.

Fig. 7 shows the RLS-corrected external wrench estimate. The estimated force components and force norm increase during bilateral spreading and decrease after bilateral contact ends, indicating sustained obstacle interaction. The dominant force component is along the body $\hat{\mathbf{y}}_b$ axis, which is the spreading direction $\hat{\mathbf{s}}_b$, and has a peak magnitude of approximately 1.56 N. The peak absolute force components are approximately 0.93 N, 1.56 N, and 1.37 N along the $\hat{\mathbf{x}}_b$, $\hat{\mathbf{y}}_b$, and $\hat{\mathbf{z}}_b$ axes, respectively, and the peak force norm is approximately 2.11 N. The peak absolute moment components are approximately $0.30\ \mathrm{N\,m}$, $0.67\ \mathrm{N\,m}$, and $0.41\ \mathrm{N\,m}$ along the $\hat{\mathbf{x}}_b$, $\hat{\mathbf{y}}_b$, and $\hat{\mathbf{z}}_b$ axes, respectively, with a peak moment norm of approximately $0.754\ \mathrm{N\,m}$.

The key quantitative result is the scale separation between the local obstacle-separation force and the resultant force transmitted to the vehicle. The inferred local force at full opening is nearly 88 N per bar, whereas the peak resultant force estimated at the vehicle is 2.11 N, corresponding to a local-to-resultant force ratio of 40. This occurs because opposing contact forces make the resultant force on the vehicle depend primarily on force imbalance rather than individual contact-force magnitudes. The spreading mechanism supplies the local force needed to displace the obstacles, and the variable admittance controller modifies the reference during interaction so that the vehicle remains bounded instead of rigidly tracking the desired reference.

5 CONCLUSION

This paper presented variable admittance control for elastic obstacle separation with a fully actuated fixed-tilt aerial manipulator. The FT-X8 uses a single-actuator spreading mechanism to apply opposing contact forces while the rotors regulate vehicle motion. The admittance law varies the virtual inertia and stiffness as functions of the estimated wrench component, obstacle distance, and actuator saturation, producing stiff free-flight tracking and compliant contact response without mode switching. In the spreading experiment, the inferred local obstacle-separation force was 40 times the peak resultant force estimated at the vehicle. This result demonstrates large local separation forces with bounded vehicle motion during contact.

ACKNOWLEDGEMENTS

The research 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] Anibal Ollero, Marco Tognon, Alejandro Suarez, Dongjun Lee, and Antonio Franchi. Past, present, and future of aerial robotic manipulators. *IEEE Transactions on Robotics*, 38(1):626–645, 2022.

[2] Fabio Ruggiero, Vincenzo Lippiello, and Anibal Ollero. Aerial manipulation: A literature review. *IEEE Robotics and Automation Letters*, 3(3):1957–1964, 2018.

[3] Pisak Chermprayong, Ketao Zhang, Feng Xiao, and Mirko Kovac. An integrated delta manipulator for aerial repair: A new aerial robotic system. *IEEE Robotics & Automation Magazine*, 26(1):54–66, 2019.

[4] Federico Augugliaro and Raffaello D'Andrea. Admittance control for physical human-quadrocopter interaction. In *2013 European Control Conference (ECC)*, pages 1805–1810, 2013.

[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] Hughes La Vigne, Guillaume Charron, Julien Rachiele-Tremblay, David Rancourt, Ben Nyberg, and Alexis Lussier Desbiens. Collecting critically endangered cliff plants using a drone-based sampling manipulator. *Scientific Reports*, 12(1):14827, 2022.

[7] Ramy Rashad, Jelmer Goerres, Ronald G. K. M. Aarts, Johan B. C. Engelen, and Stefano Stramigioli. Fully actuated multirotor uavs: A literature review. *IEEE Robotics & Automation Magazine*, 27(3):97–107, 2020.

[8] Markus Ryll, Giuseppe Muscio, Francesco Pierri, Elisabetta Cataldi, Gianluca Antonelli, Fabrizio Caccavale, Davide Bicego, and Antonio Franchi. 6d interaction control with aerial robots: The flying end-effector paradigm. *The International Journal of Robotics Research*, 38(9):1045–1062, 2019.

[9] Karen Bodie, Maximilian Brunner, Michael Pantic, Stefan Walser, Patrick Pfändler, Ueli Angst, Roland Siegwart, and Juan Nieto. Active interaction force control for contact-based inspection with a fully actuated aerial vehicle. *IEEE Transactions on Robotics*, 37(3):709–722, 2021.

[10] Chao Ding and Lu Lu. A tilting-rotor unmanned aerial vehicle for enhanced aerial locomotion and manipulation capabilities: Design, control, and applications. *IEEE/ASME Transactions on Mechatronics*, 26(4):2237–2248, 2021.

[11] Moju Zhao, Kei Nagato, Kei Okada, Masayuki Inaba, and Masaru Nakao. Forceful valve manipulation with arbitrary direction by articulated aerial robot equipped with thrust vectoring apparatus. *IEEE Robotics and Automation Letters*, 7(2):4893–4900, 2022.

[12] Pedro H. M. Souza and Karl Stol. Analysis and design of a novel fully-actuated compact multirotor. In *Proceedings of the Australasian Conference on Robotics and Automation*, 2021.

[13] Pedro Henrique Mendes Souza and Karl Stol. Constrained dynamics of an aerial manipulator interacting with flexible cantilever beams. *IEEE/ASME Transactions on Mechatronics*, 28(2):967–975, 2023.

[14] A. Q. L. Keemink, H. van der Kooij, and A. H. A. Stienen. Admittance control for physical human-robot interaction. *The International Journal of Robotics Research*, 37(11):1421–1444, 2018.

[15] 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.

[16] Gwang Kang, Hwan-Soo Oh, Jeong-Ki Seo, Uikyum Kim, and Hyouk Ryeol Choi. Variable admittance control of robot manipulators based on human intention. *IEEE/ASME Transactions on Mechatronics*, 24(3):1023–1032, 2019.

[17] Hui-Yu Li, Ishara Paranawithana, Liangjing Yang, T. S. K. Lim, Shaohui Foong, F. C. Ng, and U-Xuan Tan. Stable and compliant motion of physical human-robot interaction coupled with a moving environment using variable admittance and adaptive control. *IEEE Robotics and Automation Letters*, 3(3):2493–2500, 2018.

[18] PX4 Development Team. PX4 Autopilot Software. `https://github.com/PX4/PX4-Autopilot`. Accessed: 2026-05-15.

[19] Dan Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley-Interscience, Hoboken, NJ, USA, 2006.

[20] Jyh-Shing Roger Jang. ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(3):665–685, 1993.

http://www.imavs.org/

Online Wind Field Estimation for Airborne Wind Energy Systems in Circular Flight

Jakob Harzer[1]*, Felix Renard[1], Kim Polkläser[2], Dieter Moormann[2], Moritz Diehl[1]
[1]University of Freiburg, Germany, [2]RWTH Aachen University, Germany

http://www.imavs.org/

Abstract

Airborne Wind Energy (AWE) systems utilize tethered wings flying high-speed crosswind trajectories to harvest energy from the stronger and more consistent winds available at high altitudes. Knowledge of the local wind field is crucial for maximizing power generation while ensuring flight stability and safety, particularly because these periodic trajectories remain spatially confined. In this work, we estimate the parameters of a wind field model capturing both shear and veer during the online operation of an AWE system flying periodic circular orbits, utilizing an Extended Kalman Filter (EKF). We then use these estimates in a Nonlinear Model Predictive Control (NMPC) scheme based on a dynamic model that incorporates the spatially varying wind field. We demonstrate in a simulation study that the proposed approach accurately estimates sudden variations in the wind parameters during online closed-loop tracking while satisfying operational flight envelope constraints.

1 Introduction

Airborne wind energy (AWE) systems produce renewable electric energy by harvesting wind using tethered wings at high altitudes where, compared to traditional wind turbines, the available wind is stronger and more consistent. The systems fly acrobatic crosswind maneuvers at high speeds and thus generate large aerodynamic forces which are converted to usable electric energy using one of two approaches: Systems operating in 'drag-mode' transfer the electricity from onboard propeller generators to the ground via the tether, whereas in systems operating in 'pumping-mode', the tether drum is connected to a generator on the ground. Here, the tether is reeled out at high tether force to produce energy and reeled back in with low tether force in a periodic pumping motion. The technology promises renewable energy generation with less material investment and ground use compared to traditional wind energy [1]. In Figure 1, we show both an example of a 'soft-wing' system in pumping mode as well as a 'rigid-wing' system in drag mode. For more information on airborne wind energy we refer the reader to [4].

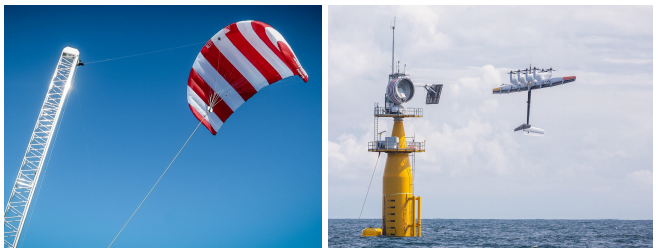


Figure 1: Left: Skysails PN-14 soft-wing system from [2], right: Makani M600 drag-mode rigid-wing system, from [3].

The tethered flight systems typically operate on flight trajectories in a limited spatial range, which are optimized based on a chosen model of the local wind field. While for planning purposes complex wind fields are considered [5], for the online closed-loop operation simpler wind field models are commonly used. We propose to estimate the fluctuating parameters of a complex model of the current wind field with altitude-dependent wind speed and direction online using an Extended Kalman Filter (EKF). To make use of the estimates, a controller with knowledge about the spatially changing wind field is required, for which we propose to use a Nonlinear Model Predictive Controller (NMPC). Equipped with an accurate wind field model, such a predictive controller is expected to have better tracking performance compared to a controller that does not know about the spatially changing wind conditions. We visualize the idea in Figure 2, where we show both the wind field used to generate the tracking reference as well as the actual wind-field whose wind direction and magnitude are altitude-dependent. The forward prediction horizon of the NMPC is visualized originating from the tethered flight system's current position.

State and wind estimation for airborne wind energy systems has been subject to prior research. Typically, the available measurements contain positions and velocities from GPS measurements, orientations, their rates, and accelerations from an IMU, airspeed from Pitot tubes as well as the tether force and angle. With well-established methods such as EKF, unscented Kalman filters or moving-horizon estimation, these measurements can be fused into state and wind estimates [6–8]. We refer the reader to Table 1 in [6] for an overview. The wind field estimate typically consists of

*Corresponding author email: jakob.harzer@imtek.uni-freiburg.de. This research was supported by DFG via projects 504452366 (SPP 2364), 560056112 (robust MPC), 535860958 (ALeSCo), 525018088 (MAWERO) and 545539514, and by BMWK via 03EN3054B.

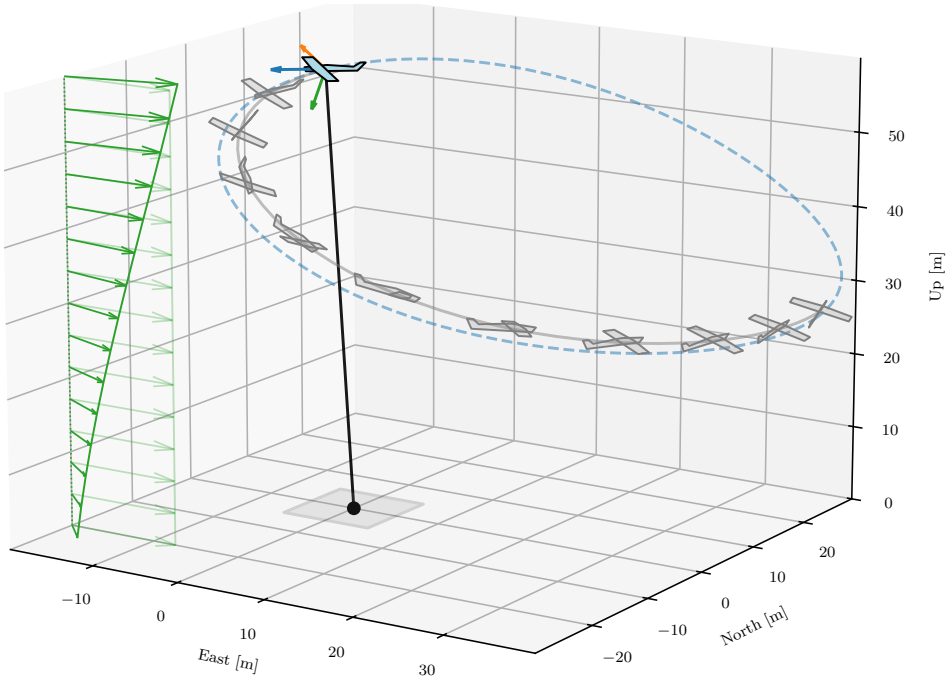


Figure 2: The closed-loop NMPC controller uses an EKF estimate of the actual wind field that might differ from the one used for the reference trajectory generation

wind direction and magnitude [9, 10] or parameters of a one-dimensional logarithmic wind-shear model with known direction [11]. In [6], the wind speed and direction variability in space and time is handled by assuming appropriate process noise on the parameter estimates. Such estimates of the wind field parameters can be used to adjust the periodic flight orbit to changing conditions [11, 12], but require a dynamic model of the system [10]. Particularly suited for the inclusion of complex wind models are NMPC control schemes which offer out-of-the-box satisfaction of system constraints, but only have been used in combination with simple wind models [7, 8]. This work attempts to bridge this gap by estimating the parameters of a wind field with both shear and veer and utilizing the estimation in a predictive controller.

We introduce an AWE system model in Section 2 and describe the closed-loop control and wind-state estimation in Section 3. We showcase the performance of the approach in Section 4 and conclude the paper with Section 5.

2 System Model

In the following, we develop a model to describe the tethered rigid-wing AWE system with RWTH Aachen's "Maverix" flight system. The Maverix is a versatile, multi-purpose flight-system with a wingspan of $1.5\,\mathrm{m}$ and a mass of $2.4\,\mathrm{kg}$. Figure 3 shows a render of the flight system. In this paper, we consider it to be connected to the origin with a tether of constant length $l = 50\,\mathrm{m}$. The flight system has propellers on both its wings. Although in an AWE setup we do not want to spend additional energy for propulsion, we still include the propeller thrust in the dynamic model and also utilize them in the reference solution and for feedback control. The Maverix flight system also has vertical liftoff and landing capabilities, which are disregarded in the model.

2.1 State-Space Model

The following is largely based on [5, 13]. We model the AWE system with the state vector

$$x = [r, \dot{r}, R, \omega, \xi, \eta, \sigma_r, \sigma_l]^\top \in \mathbb{R}^{22} \tag{1}$$

where $r = [r_x, r_y, r_z]^\top \in \mathbb{R}^3$ is the position of the flight system in a north-east-down (NED) inertial frame (e_x, e_y, e_z), $\dot{r} \in \mathbb{R}^3$ is its velocity in the same NED inertial frame, $R \in \mathrm{SO}(3) \subset \mathbb{R}^9$ is a direction-cosine rotation matrix describing the attitude and $\omega \in \mathbb{R}^3$ is the vector of angular velocities in the NED body frame. The state vector is extended with actuator states for the ailerons on the left and right wings, which are assumed to be actuated symmetrically with angle $\xi = \xi_l = -\xi_r$, an elevator flap angle η and left and right propellors' forward thrusts σ_r, σ_l. The considered control vector is then simply given as the rates of the actuator states:

$$u = [\dot{\xi}, \dot{\eta}, \dot{\sigma}_r, \dot{\sigma}_l] \in \mathbb{R}^4 \tag{2}$$

Following [5, 7, 13], we derive the dynamic model of the tethered AWE system using Lagrangian dynamics. For this,

http://www.imavs.org/



Figure 3: Render of RWTH Aachen's Maverix flight system

we introduce an algebraic variable $z \in \mathbb{R}$ that is proportional to the tether pulling force with magnitude $|F_\mathrm{t}| = zl$. The system dynamics in fully implicit form are then:

$$0 = f(\dot{x}, x, z, u, p) \tag{3}$$

where the vector $p \in \mathbb{R}^{n_p}$ contains the parameters of the model, in this case explicitly the coefficients of the wind field discussed below. In contrast to [5, 7, 13], for simplicity we omit tether drag forces, and the tether is assumed to be of constant length. The model has the algebraic tether constraint $c(x) = r^\top r - l^2 = 0$ which implies a "rigid" tether and requires $z > 0$ for validity. Since with this algebraic state constraint, the dynamics would be an index-3 differential-algebraic equation (DAE), we apply index reduction to modify the dynamics into an index-1 DAE whose solution satisfies the tether and rotation matrix invariants:

$$c(x) = 0, \quad \dot{c}(x) = 0, \quad c_R(x) = P_\mathrm{u}\big(R^\top R - \mathbb{I}_3\big) = 0, \tag{4}$$

where the operator P_u selects the upper triangular elements of the matrix. Thus, at the initial state of a numerical simulation, these consistency conditions have to be satisfied explicitly. To avoid drifting of the invariants due to numerical instability, we use a Baumgarte stabilization scheme. We refer the reader to [5, 13] for further details on the exact model dynamics.

2.2 Wind Model

We assume a wind field where the wind: (a) only changes with altitude, and (b) does not have a vertical component (i.e., the wind is always horizontal). In particular, we consider the following model that captures both wind shear and wind veer:

$$v_w(r_z, p) = \begin{bmatrix} \cos(p_1 + p_4(r_z - z_0)) \\ \sin(p_1 + p_4(r_z - z_0)) \\ 0 \end{bmatrix} (p_2 + p_3(r_z - z_0)) \tag{5}$$

where the four scalar parameters $p = [p_1, p_2, p_3, p_4]^\top$ take the meaning of the wind direction p_1 and speed p_2 at a reference height z_0 and the gradients p_4, p_3 of the direction and the speed with respect to the altitude r_z, respectively.

2.3 Aerodynamics

To compute the aerodynamic forces and moments for the particular flight system considered in this paper we use an element-based simulation model which divides the aircraft into individual aerodynamic surfaces. The simulation computes the force and moment vectors for each element for a given local airspeed, angular velocity and actuator deflection and calculates the total forces and moments affecting the system in the body-fixed coordinate frame. This element-based method includes aerodynamic effects such as induced drag, lift, and propeller slipstream and is suited for a tilt-wing aircraft, as is considered in this paper. An detailed description of the element-based modeling method can be found in [14].

For the flight path trajectory optimization, in particular for the online predictive control scheme, we require a differentiable low-fidelity model. For the given flight system, we obtain this low-fidelity model by sampling the 11-dimensional input space of the simulation model over a large range of flight states to capture the complex nonlinear model dynamics within a simplified data set. This data set contains approximately 13 million data points, each representing a different operating point. Subsequently, we fit a low-order polynomial onto the data where we choose the 71 polynomial regressors such that they include known aerodynamic quantities, for example the squared airspeed or the angle of attack. Since the model should require as few computations as possible, we solve a sparse linear least-squares fitting problem for each individual component of the forces and moments vectors using an extended Gauss-Newton method [15]. The polynomial approximations with up to 35/71 non-zero coefficients show a good fit on the 20% test split of the data with a standard deviation of the residuals less than 4 % for each component.

2.4 System Constraints

We have tight constraints on the flight envelope of the Maverix flight system, and thus require the satisfaction of nonlinear and linear state and control constraints given in Table 1. Here,

$$v_\mathrm{b} = R^\top(\dot{r} - v_w(r_z, p)) \tag{6}$$

is the velocity vector in the body-fixed coordinate system relative to the airflow, $\alpha = \mathrm{atan2}(v_{\mathrm{b},z}, v_{\mathrm{b},x})$ is the angle of attack, $\beta = \mathrm{atan2}(v_{\mathrm{b},y}, v_{\mathrm{b},x})$ is the sideslip angle and $\gamma = \cos^{-1}(-l^{-1} r^\top R e_z)$ is the angle between the tether vector and the flight systems upward orientation unit vector, which we constrain to avoid the collision of the flight system with the tether. We collect all nonlinear and linear state and control constraints as

$$h_\mathrm{min} \le h(x, z, p) \le h_\mathrm{max} \tag{7a}$$
$$x_\mathrm{min} \le x \le x_\mathrm{max} \tag{7b}$$
$$u_\mathrm{min} \le u \le u_\mathrm{max} \tag{7c}$$

http://www.imavs.org/

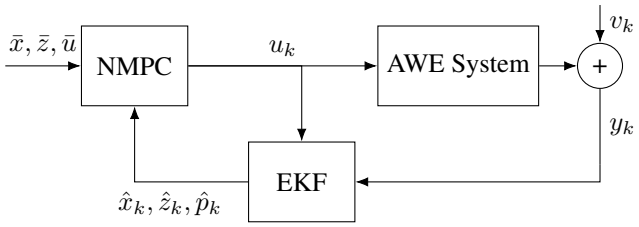


Figure 4: Closed-loop simulation

2.5 *Measurements*

We assume that the flight system is equipped with sensors that provide the measurements

$$g(x,z,p) = \begin{bmatrix} r \\ \dot{r} \\ \omega \\ \|v_\mathrm{b}\|_2 \\ R^\top(\ddot{r} - a_g) \\ F_t \end{bmatrix} \tag{8}$$

including an GPS measurement of position and velocity, an IMU measurement of angular velocity in the body frame, an airspeed measurement, an IMU measurement of acceleration in the body frame, as well as a measurements of the tether force. Here, $a_g = [0,0,9.81]^\top \mathrm{m/s^2}$ is the gravitational acceleration in the NED inertial frame. The measurements are summarized in Table 2 where we follow [7] for their standard deviation values, except for the force measurement where we assume a standard deviation of 1 N.

3 CLOSED-LOOP WIND PARAMETER ESTIMATION

In order to estimate the wind parameters p in online operation of the tethered flight system, we consider the following closed-loop approach visualized in Figure 4. The available noisy measurements from the system, as well as the applied control u_k, are used by an Extended Kalman Filter (EKF) to obtain both a state estimate $\hat{x}_k, \hat{z}_k$ and an estimate for the wind parameters $\hat{p}_k$. A Nonlinear Model Predictive Controller (NMPC) generates a feedback control signal such that the tethered system follows a pre-computed reference state and control trajectories $\bar{x}, \bar{z}, \bar{u}$. We assume that we obtain a measurements every $\delta_1 = 10\,\mathrm{ms}$, for which we obtain a new state estimate from the EKF. The NMPC is sampled every $\delta_2 = 10\delta_1 = 100\,\mathrm{ms}$ and predicts $N = 50$ intervals into the future, which corresponds to a horizon of $N\delta_2 = 5\,\mathrm{s}$.

3.1 *Model Discretization*

We discretize the continuous model as follows: For a discrete time point t_k on a time-grid with step size δ, consider the approximations $x_k \approx x(t_k), z_k \approx z(t_k)$. We assume that over a discretization interval $t \in [t_k, t_k + \delta]$, the control $u(t) = u_k$ is held constant. The fully implicit index-1 differential-algebraic equation system (3) is integrated using an implicit Runge-Kutta scheme with a collocation Gauss-Radau IIA method which is well-suited for the stiff dynamics. Since the last of the three collocation nodes is at the end of the interval, the algebraic state z_{k+1} is uniquely determined from the state values and does not depend on z_k. We then have the discrete deterministic model:

$$0 = F_\delta(x_{k+1}, z_{k+1}, x_k, u_k, p), \tag{9}$$

and, by incorporating measurements, the discrete stochastic model:

$$0 = F_\delta(x_{k+1}, z_{k+1}, x_k, u_k, p) \tag{10a}$$
$$y_k = g(x_k, z_k, p) + v_k \tag{10b}$$

Here, the measurement noise is $v_k \sim \mathcal{N}(0, V)$ for whose covariance matrix V is built from the standard deviations given in Table 2. Note that we do not consider any state noise.

3.2 *Reference Trajectory Generation*

For the tracking reference, we generate dynamically feasible periodic state and control trajectories for the system by formulating and solving an optimal control problem with direct collocation where we use the simple constant wind-model with $p_1 = 90°, p_2 = 5\,\mathrm{m/s}, p_3 = 0/\mathrm{s}, p_4 = 0\,°/\mathrm{m}$. We obtain the T-periodic references:

$$\bar{x} : \mathbb{R} \to \mathbb{R}^{n_x}, \qquad \bar{x}(t+T) = \bar{x}(t), \tag{11a}$$
$$\bar{z} : \mathbb{R} \to \mathbb{R}^{n_z}, \qquad \bar{z}(t+T) = \bar{z}(t), \tag{11b}$$
$$\bar{u} : \mathbb{R} \to \mathbb{R}^{n_u}, \qquad \bar{u}(t+T) = \bar{u}(t). \tag{11c}$$

Typically, we want the AWE system to follow power-optimal flight trajectories [5], but the considered experimental proof-of-concept system based on the Maverix flight system is not intended to generate energy. Thus, we consider the objective of a large, constant tether force as proxy for the objective of producing power. We then design the optimal control problems cost accordingly such that we obtain a smooth circular trajectory with radius $\approx 50\,\mathrm{m}$ that is tilted 30° to the east, with almost constant tether force of $\approx 50\,\mathrm{N}$ and a period duration of $T \approx 11\,\mathrm{s}$. We use slightly tightened aerodynamic bounds to account for disturbances in the later online tracking.

3.3 *Model Predictive Control Scheme*

To exploit the wind field estimates in closed-loop operation, we employ an NMPC scheme equipped with the discretized dynamical model (9). At any time t_0, we have the NMPC horizon as the uniform time grid: $t_k = t_0 + k\delta_2, k = 1, \ldots, N$ with $N = 50$ intervals. Note that this grid is different from the simulation time grid above. The controller tries to track the pre-calculated reference trajectories by sampling them as:

$$\bar{x}_k = \bar{x}(t_k), \qquad \bar{z}_k = \bar{z}(t_k), \qquad \bar{u}_k = \bar{u}(t_k), \tag{12}$$

http://www.imavs.org/

where the tracking cost terms are weighted by

$$Q = \mathrm{diag}(Q_x, Q_z, Q_u), \tag{13a}$$
$$Q_\mathrm{f} = \mathrm{diag}(Q_x, Q_z), \tag{13b}$$

where the values of the diagonal matrices, given in Table 4, are chosen to emphasize the tracking of the tether force vector with small deviations in angular rates and flap controls. For a given state and parameter estimate $(\hat{x}, \hat{z}, \hat{p})$ at some time t_0, we then solve the NMPC problem:

$$\min_{\substack{x_0,\dots,x_N \\ z_0,\dots,z_N \\ u_0,\dots,u_{N-1}}} \quad \sum_{k=0}^{N-1} \left\| \begin{matrix} x_k - \bar{x}_k \\ z_k - \bar{z}_k \\ u_k - \bar{u}_k \end{matrix} \right\|_Q^2 + \left\| \begin{matrix} x_N - \bar{x}_N \\ z_N - \bar{z}_N \end{matrix} \right\|_{Q_\mathrm{f}}^2 \tag{14a}$$

$$\text{s.t.} \quad x_0 = \hat{x}, \tag{14b}$$
$$z_0 = \hat{z}, \tag{14c}$$
$$0 = F_{\delta_2}(x_{k+1}, x_k, z_{k+1}, u_k, \hat{p}), \quad \forall k \in \mathcal{K}_{N\text{-}1}, \tag{14d, 14e}$$
$$u_\mathrm{min} \le u_k \le u_\mathrm{max}, \quad \forall k \in \mathcal{K}_{N\text{-}1}, \tag{14f}$$
$$x_\mathrm{min} \le x_k \le x_\mathrm{max}, \quad \forall k \in \mathcal{K}_N, \tag{14g}$$
$$h_\mathrm{min} \le h(x_k, z_k, \hat{p}) \le h_\mathrm{max}, \forall k \in \mathcal{K}_N \tag{14h}$$

where $\mathcal{K}_N = \{0, 1, \dots, N\}$ and $\mathcal{K}_{N-1}$ are index sets and we re-introduce the discrete states x_k, z_k, u_k for the NMPC problem. The control and state constraints are enforced as hard constraints, while the nonlinear constraints are implemented as soft constraints with slack variables and an associated penalty to improve problem feasibility. We solve problem (14) with the fast embedded NMPC solver acados [16] in combination with the CasADi [17] framework, where we use the advanced-step real-time iteration scheme [18] with a single sequential quadratic programming (SQP) step per sampling instant. The solver is configured with a Gauss–Newton Hessian approximation and a partial-condensing quadratic-program solver "HPIPM" [19]. The first computed control u_0 is then returned to the plant as the closed-loop feedback.

3.4 State and Wind Parameter Estimation

The dynamics of the system (3) contain algebraic conditions as well as consistency conditions that need to be satisfied by the state estimates, for which a conventional EKF implementation is not suitable. Instead, we follow [20] for details on EKF estimation with index-1 DAE systems. For the estimation of both the state and the wind field parameters, it is convenient to augment the state vector and dynamics with a parameter state $\dot{p} = 0$ as $x^a = [x,\, p]^\top \in \mathbb{R}^{n_x+n_p}$. Additionally, we can explicitly split the fully implicit index-reduced DAE into differential and algebraic parts. We then obtain the model in semi-explicit form:

$$\dot{x}^a = f^a_\mathrm{ode}(x^a, z, u), \tag{15a}$$
$$0 = f_\mathrm{alg}(x^a, z, u). \tag{15b}$$

The state update based on the linearization of the function g might violate the consistency conditions. To counteract this, we also augment the measurements with a pseudo-measurement of the consistency conditions following [21], similar to [22]:

$$y^a = g^a(x, z) = \begin{bmatrix} g(x, z, p) \\ c(x) \\ \dot{c}(x) \\ c_R(x) \end{bmatrix} \tag{16}$$

For the EKF prediction step, we propagate the current (augmented) estimates $\hat{x}^a_{k|k}, \hat{z}_{k|k}$ of the differential and algebraic states using the implicit Runge-Kutta integration method on the dynamics (15) similar to Equation (9) to obtain predicted $\hat{x}^a_{k+1|k}, \hat{z}_{k+1|k}$. In order to propagate the uncertainty covariances of both the augmented differential state and algebraic state $P_k \in \mathbb{R}^{(n_x+n_p+n_z)\times(n_x+n_p+n_z)}$, we linearize the semi-explicit DAE at the current estimate:

$$A = \begin{bmatrix} \frac{\partial f_\mathrm{ode}}{\partial x^a} & \frac{\partial f_\mathrm{ode}}{\partial z} \\ \frac{\partial f_\mathrm{alg}}{\partial x^a} & \frac{\partial f_\mathrm{alg}}{\partial z} \end{bmatrix}_{\hat{x}^a_{k|k}, \hat{z}_{k|k}, u_k} \tag{17}$$

and with $\phi = \exp(\Delta t A)$ we propagate the covariance matrices as:

$$P_{k+1|k} = \phi P_{k|k} \phi^\top + \Gamma W \Gamma^\top \tag{18}$$

where

$$\Gamma = \begin{bmatrix} \mathbb{I}_{n_x+n_p} \\ -\left(\frac{f_\mathrm{alg}}{\partial z}\right)^{-1} \frac{\partial f_\mathrm{alg}}{\partial x^a} \end{bmatrix}_{\hat{x}^a_{k|k}, \hat{z}_{k|k}, u_k} \tag{19}$$

and where W is the covariance matrix of the augmented state noise with values from Table 3. For the EKF update step, we correct the differential state estimate

$$\begin{bmatrix} \hat{x}^a_{k+1|k+1} \\ \hat{z}_{k+1|k+1} \end{bmatrix} = \begin{bmatrix} \hat{x}^a_{k+1|k} \\ \hat{z}_{k+1|k} \end{bmatrix} \tag{20}$$
$$+ K_{k+1} \left(y^a_{k+1} - g^a(\hat{x}^a_{k+1|k}, \hat{z}_{k+1|k}) \right) \tag{21}$$

with the measurement augmented with the zero pseudo-measurement for the consistency conditions as $y^a_{k+1} = [y_{k+1}, 0, \dots, 0]^\top$. The covariance matrix is updated as

$$P_{k+1|k+1} = (\mathbb{I} - K_{k+1} G_{k+1}) P_{k+1|k} \tag{22}$$

based on the Kalman gain

$$K_{k+1} = \left(P^{-1}_{k+1|k} + G^\top_{k+1} V_k^{-1} G^\top_{k+1} \right)^{-1} G^\top_{k+1} V_k^{-1} \tag{23}$$

with the linearization of the measurement function $G_{k+1} = \frac{\partial g^a}{\partial (x^a, z)}(\hat{x}^a_{k+1|k}, \hat{z}_{k+1|k})$.

http://www.imavs.org/

http://www.imavs.org/

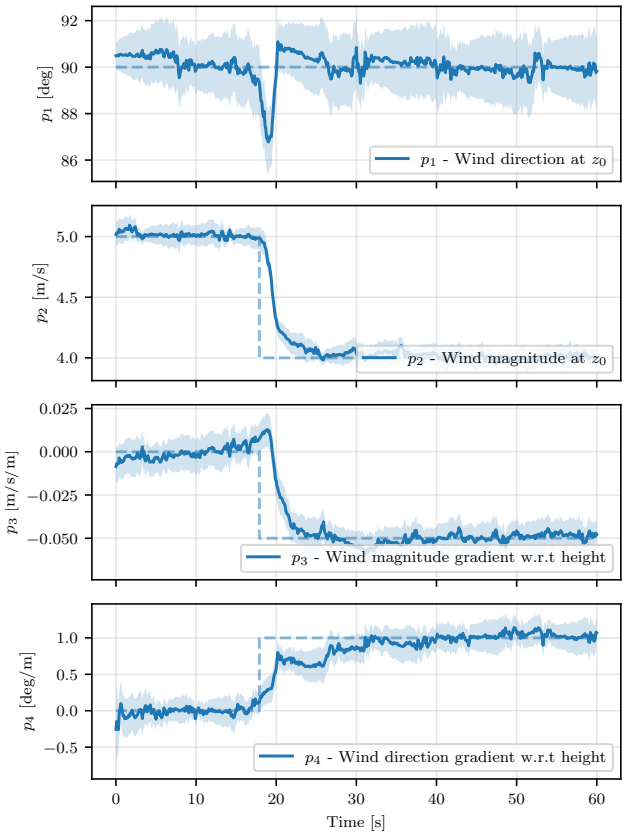


Figure 5: Convergence of the wind parameter estimates during the closed-loop simulation

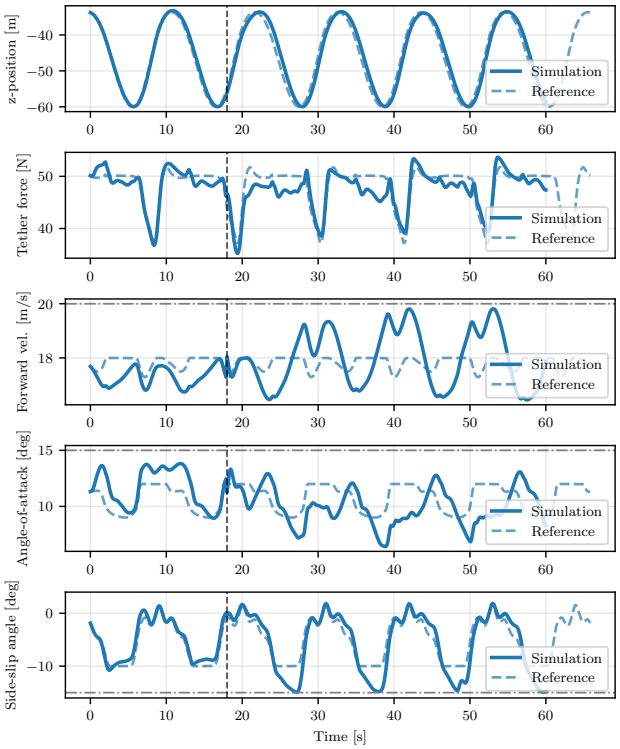


Figure 6: Comparison between implemented simulation and the tracking reference

4 Simulation Experiments

To demonstrate the approach, we perform a simulation of the closed-loop system where the tethered flight system should follow the generated circular tilted reference. In the scenario, the wind conditions are the same as in the reference generation at first, but after 18 s, the conditions jump to a more complex wind field:

$$p(t) = \begin{cases} [90°, 5\,\mathrm{m/s}, 0/\mathrm{s}, 0\,°/\mathrm{m}]^\top & \text{for } t < 18\,\mathrm{s} \\ [100°, 4\,\mathrm{m/s}, -0.06/\mathrm{s}, 1\,°/\mathrm{m}]^\top & \text{for } t \geq 18\,\mathrm{s} \end{cases} \tag{24}$$

where the wind speed drops by $1\,\mathrm{m/s}]$ and there is now both a gradient in the speed and in the direction. We draw an initial state estimate from distribution: $\hat{x}_0^a \sim \mathcal{N}(x_0^a, P_0)$ where the EKF's initial covariance matrix P_0 is given from the value in Table 3, and the simulation's initial value is given by $x_0^a = [\bar{x}(0), p(0)]^\top$.

Figure 5 shows the wind field parameter estimates, which closely follow the true values. Figure 6 compares the simulation with the tracked reference trajectory. The state that are supposed to be tracked by the NMPC (e.g.z-position and tether force) closely follow the reference. In the first few seconds after the wind field changes, the wind-field estimate is not converged yet. Since the tether force is very sensitive to the wind conditions, the NMPC struggles to maintain the tracked tether forces (second plot in Figure 6) at first but later compensates for the wind conditions by increasing the peak forward velocity (third plot). After the wind field estimates have converged to the new conditions, the nonlinear aerodynamic envelope constraints remain satisfied (bottom three plots in Figure 6), even though perfect tracking of the reference is no longer achieved.

5 Conclusion

In this paper, we outlined a closed-loop estimation approach of the parameters of a shear and veer wind field model for an airborne wind energy system operating in periodic circular orbits. In a simulation of a proof-of-concept setup with the 'Maverix' flight system, we can accurately estimate the wind field parameters. The NMPC scheme utilizes these estimates to satisfy nonlinear flight envelope constraints while still tracking the reference trajectory.

Acknowledgments We thank Johannes Schuler, Bagasprawira Herlambang, Katrin Baumgärtner, and Jonathan Frey for their support during the writing process. For this publication, large language models were used for code generation, literature research, and grammar checks.

The authors claim full responsibility for the contents of the publication.

REFERENCES

[1] Jochem De Schutter, Jakob Harzer, and Moritz Diehl. Vertical Airborne Wind Energy Farms with High Power Density per Ground Area based on Multi-Aircraft Systems. *European Journal of Control*, 74:100867, November 2023.

[2] SkySails Power GmbH. Skysails power gmbh website. https://skysails-power.com/, 2024. Accessed: 2024-11-13.

[3] Roland Schmehl and Oliver Tulloch, editors. *8th International Airborne Wind Energy Conference (AWEC 2019): Book of Abstracts*. Delft University of Technology / University of Strathclyde, 2019.

[4] Uwe Ahrens, Moritz Diehl, and Roland Schmehl, editors. *Airborne Wind Energy*. Green Energy and Technology. Springer Berlin, Heidelberg, 2013.

[5] J. De Schutter, R. Leuthold, T. Bronnenmeyer, E. Malz, S Gros, and M. Diehl. AWEbox: An optimal control framework for single- and multi-aircraft airborne wind energy systems. *Energies*, 16(4):1900, 2023.

[6] Oriol Cayon, Simon Watson, and Roland Schmehl. Kite as a Sensor: Wind and State Estimation in Tethered Flying Systems, January 2025.

[7] Mario Zanon, Greg Horn, Sebastien Gros, and Moritz Diehl. Control of Dual-Airfoil Airborne Wind Energy systems based on nonlinear MPC and MHE. In *2014 European Control Conference (ECC)*, pages 1801–1806, Strasbourg, France, June 2014. IEEE.

[8] Sébastien Gros, Mario Zanon, Milan Vukov, and Moritz Diehl. Nonlinear MPC and MHE for Mechanical Multi-Body Systems with Application to Fast Tethered Airplanes. *IFAC Proceedings Volumes*, 45(17):86–93, 2012.

[9] Paul Williams, Bas Lansdorp, and Wubbo Ockels. Nonlinear Control and Estimation of a Tethered Kite in Changing Wind Conditions. *Journal of Guidance, Control, and Dynamics*, 31(3):793–799, May 2008.

[10] Maximilian Ranneberg. Sensor Setups for State and Wind Estimation for Airborne Wind Energy Converters, September 2013.

[11] Aldo U. Zgraggen, Lorenzo Fagiano, and Manfred Morari. Real-Time Optimization and Adaptation of the Crosswind Flight of Tethered Wings for Airborne Wind Energy. *IEEE Transactions on Control Systems Technology*, 23(2):434–448, March 2015.

[12] Mitchell K. Cobb, Kira Barton, Hosam Fathy, and Chris Vermillion. Iterative Learning-Based Path Optimization for Repetitive Path Planning, With Application to 3-D Crosswind Flight of Airborne Wind Energy Systems. *IEEE Transactions on Control Systems Technology*, 28(4):1447–1459, July 2020.

[13] Sébastien Gros and Moritz Diehl. Modeling of Airborne Wind Energy Systems in Natural Coordinates. In Uwe Ahrens, Moritz Diehl, and Roland Schmehl, editors, *Airborne Wind Energy*, pages 181–203. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[14] Philipp Hartmann. Vorausschauende Flugbahnregelung für Kippflügelflugzeuge. Deutscher Luft- und Raumfahrt Kongress 2018, Friedrichshafen (Germany), 4 Sep 2018 - 6 Sep 2018, September 2018.

[15] Katrin Baumgartner and Moritz Diehl. The Extended Gauss-Newton Method for Nonconvex Loss Functions and its Application to Time-Optimal Model Predictive Control. In *2022 American Control Conference (ACC)*, pages 4973–4978, Atlanta, GA, USA, June 2022. IEEE.

[16] Robin Verschueren, Gianluca Frison, Dimitris Kouzoupis, Jonathan Frey, Niels van Duijkeren, Andrea Zanelli, Branimir Novoselnik, Thivaharan Albin, Rien Quirynen, and Moritz Diehl. acados – a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, pages 147–183, 2021.

[17] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. 11(1):1–36, 2019.

[18] Jonathan Frey, Armin Nurkanović, and Moritz Diehl. Advanced-Step Real-Time Iterations With Four Levels – New Error Bounds and Fast Implementation in acados. *IEEE Control Systems Letters*, 8:1703–1708, 2024.

[19] Gianluca Frison and Moritz Diehl. HPIPM: A high-performance quadratic programming framework for model predictive control. *IFAC-PapersOnLine*, 53(2):6563–6569, 2020.

[20] Parham Mobed, Sudhakar Munusamy, Debangsu Bhattacharyya, and Raghunathan Rengaswamy. State and Parameter Estimation in Distributed Constrained Systems. 1. Extended Kalman Filtering of a Special Class of Differential-Algebraic Equation Systems. *Industrial & Engineering Chemistry Research*, 56(1):206–215, January 2017.

[21] D. Simon. Kalman filtering with state constraints: A survey of linear and nonlinear algorithms. *IET Control Theory & Applications*, 4(8):1303–1318, August 2010.

http://www.imavs.org/

[22] Eduardo Schmidt, Marcelo De Lellis Costa De Oliveira, Ramiro Saraiva Da Silva, Lorenzo Fagiano, and Alexandre Trofino Neto. In-Flight Estimation of the Aerodynamics of Tethered Wings for Airborne Wind Energy. *IEEE Transactions on Control Systems Technology*, 28(4):1309–1322, July 2020.

A TABLES

Table 1: Nonlinear and linear state and control constraints

Constraint - Nonlinear	Min	Quantity	Max
Longitudinal vel.	$12\,\mathrm{m/s}$	$v_{\mathrm{b},x}$	$20\,\mathrm{m/s}$
Lateral / vertical vel.	$-7\,\mathrm{m/s}$	$v_{\mathrm{b},y}, v_{\mathrm{b},z}$	$7\,\mathrm{m/s}$
Angle of attack	$-8°$	α	$15°$
Sideslip angle	$-15°$	β	$15°$
Tether force	$0\,\mathrm{N}$	$\lvert F_{\mathrm{t}}\rvert$	–
Tether collision	-	γ	$90°$
Constraint - Linear	**Min**	**Quantity**	**Max**
Angular velocities	$-1\,\mathrm{rad/s}$	$\omega_x, \omega_y, \omega_z$	$1\,\mathrm{rad/s}$
Aileron angle ξ	$-25°$	ξ	$25°$
Elevator angle η	$-32°$	η	$32°$
Thrust forces	$0\,\mathrm{N}$	σ_r, σ_l	$20\,\mathrm{N}$
Elev./Ail. rates	$-25\,°/\mathrm{s}$	$\dot{\xi}, \dot{\eta}$	$25\,°/\mathrm{s}$
Thrust force rates	$-5\,\mathrm{N/s}$	$\dot{\sigma}_r, \dot{\sigma}_l$	$5\,\mathrm{N/s}$

Table 2: Available Measurements, comp. [7]

Measurement	Quantity	σ	Unit
GPS position	r	0.1	m
GPS velocities	$\dot{r}$	0.6	m/s
Angular velocities	ω	0.02	rad/s
Airspeed	$\lVert v_{\mathrm{b}}\rVert_2$	1	m/s
Accelerations	$R^\top(\ddot{r} - a_g)$	0.05	m/s^2
Tether Force	$\lvert F_{\mathrm{t}}\rvert$	1	N

Table 3: EKF State Noise Weight and Initial Covariance Values

State	Process Noise σ	Init. Cov. σ	Unit
r	0.01	1	m
$\dot{r}$	0.1	0.5	m/s
R	0.01	0.1	-
ω	0.1	0.05	rad/s
ξ, η	0.1	0.1	rad
$\sigma_{\mathrm{r}}, \sigma_{\mathrm{l}}$	0.1	0.05	N
p_1	0.01	0.01	m
p_2	0.05	0.1	m/s
p_3	0.005	0.01	m/(s m)
p_4	0.002	0.01	rad/s

Table 4: NMPC State and Control Weights

State/Control	Value	Unit
r	0.001	$1/\mathrm{m}^2$
$\dot{r}$	0.1	$\mathrm{s}^2/\mathrm{m}^2$
R	1	-
ω	50	$\mathrm{s}^2/\mathrm{rad}^2$
ξ, η	0.001	$1/\mathrm{rad}^2$
$\sigma_{\mathrm{r}}, \sigma_{\mathrm{l}}$	0.001	$1/\mathrm{N}^2$
z	50	$\mathrm{m}^2/\mathrm{N}^2$
$\dot{\xi}$	1	$\mathrm{s}^2/\mathrm{rad}^2$
$\dot{\eta}$	1	$\mathrm{s}^2/\mathrm{rad}^2$
$\dot{\sigma}_{\mathrm{r}}$	100	$\mathrm{s}^2/\mathrm{N}^2$
$\dot{\sigma}_{\mathrm{l}}$	100	$\mathrm{s}^2/\mathrm{N}^2$

http://www.imavs.org/

Sequential Estimation of Fixed-Wing UAV Aerodynamic Coefficients: Evaluation on Pitch/Roll Flight Data

A. Kempf[a, b], E. Roussel[a], G. Strub[a], J. Farnault[a], D. Delmas[a] and E. Laroche[b]
[a]French-German Research Institute of Saint-Louis, France
E-mail: {aurelie.kempf, emmanuel.roussel, guillaume.strub}@isl.eu
[b]University of Strasbourg, France
E-mail: laroche@unistra.fr

http://www.imavs.org/

ABSTRACT

This paper presents a nonlinear system identification method for fixed-wing Unmanned Aerial Vehicles (UAVs) and its application to the estimation of aerodynamic coefficient derivatives of a Vertical Take-Off and Landing (VTOL) drone. The approach relies on a physics-based nonlinear model of the system and requires only a few dozen seconds of flight dataset, thereby minimizing the experimental burden. The identification process is based on the output-error method, applied sequentially to several sub-models. Hence, the aerodynamic coefficients are estimated in stages, which eliminates strong parameter interdependence and enhances model identifiability. The results obtained on the VTOL prototype are consistent with coefficient values predicted from Computational Fluid Dynamics simulations, and provide a satisfactory fit to the flight data. The methodology is generic and can be applied to other nonlinear aerial systems.

1 INTRODUCTION

Systems such as Vertical Take Off and Landing (VTOL) aircraft have attracted increasing interest over the last two decades, but come with complex aerodynamic behavior [1, 2]. Consequently, a controller that can handle wide-ranging changes in the whole flight envelope is needed. Designing such a controller requires an accurate aerodynamic model.

While Computational Fluid Dynamics (CFD) simulations and wind tunnel tests can provide high-fidelity estimations of the steady-state aerodynamic coefficients [3, 4], these methods are expensive and time-consuming. Moreover, estimating dynamic coefficients requires unsteady flow conditions impractical for typical wind tunnel setups, or coupled CFD/dynamics that are often computationally prohibitive. System identification from flight tests provides an efficient data-driven alternative to estimate the aerodynamic coefficients [5, 6, 7, 8]. Existing system identification approaches can broadly be categorized into neural-network-based, frequency-domain, and time-domain methods. Artificial neural networks are attractive when first-principles models are difficult to obtain [9]. Although effective for capturing complex nonlinear UAV dynamics, they produce black-box models with limited physical interpretability. System identification can also be performed in the frequency domain, using methods such as output-error, equation error or empirical transfer function estimate [10]. These methods are generally computationally efficient, although the transformation from time to frequency domain may result in information loss. In the time domain, recursive parameter estimation techniques enable online estimation using algorithms such as recursive least squares, recursive output-error and augmented Kalman filtering. These recursive methods are well suited for time-varying parameter tracking. Nevertheless, their slower convergence and potentially lower final accuracy make offline methods preferable for high-precision model development. For instance, filter-error methods, which typically rely on Kalman filtering [11], account for both process and measurement noise, but require higher computational cost. Equation error methods rely on least-squares estimation, which consists in minimizing a cost function defined in terms of an input-output equation [12]. Recent advances include sparse identification methods (SINDy), which use L1-regularized regression on candidate function libraries to automatically discover parsimonious nonlinear models from measured states [13]. These techniques are appreciated for their mathematical simplicity. However, all the system states and their time derivatives need to be measured, which is not always possible. Finally, the output-error method minimizes prediction errors between measured and simulated outputs obtained from model integration. It can handle nonlinear systems of arbitrary complexity, although they typically model uncertainty only through measurement noise.

Unlike traditional approaches that separate only longitudinal and lateral-directional dynamics [10, 11, 12], a method that estimates aerodynamic derivatives using more steps is proposed in [14]. This approach involves four steps: 1/ an initial polynomial fit for lift, drag, and moment coefficients and identification of longitudinal model; 2/ identification of lateral model; 3/ combined estimation of remaining coefficients; 4/ a final simultaneous adjustment of all derivatives. To obtain their results, the authors have performed about 400 maneuvers. In contrast, this paper presents a similar physics-based nonlinear identification method applied to a simpler model,

http://www.imavs.org/

enabling to use fewer datasets. It achieves satisfactory results for a VTOL UAV in fixed-wing mode by sequentially applying the output-error method to seven decoupled submodels.

The remaining sections of this article are organized as follows. Section 2 provides an overview of the VTOL drone that is used in this study. Then, Section 3 details the system modeling, and Section 4 derives the nonlinear identification method. Finally, Section 5 concludes the paper.

2 SYSTEM OVERVIEW

The proposed identification method has been evaluated on the VTOL tail-sitter UAV that is presented in Figure 1. It has four control surfaces at the wing's trailing edge; two are operating as elevons and two as rudderons. They are controlled using three servomotors (two for the elevons, one for the rudder). The UAV is powered by a single engine located in the middle of the wing. The powertrain is composed of a Hacker A30-16M Brushless DC motor, an APC 8×8E propeller and a Dualsky Summit 60X U8 Electronic Speed Controller (ESC). The avionics are centered around a mRO Control Zero H7 Pixhawk-compatible autopilot running PX4 [15]. Besides the IMU and barometer onboard the autopilot, a Pitot tube is fitted in the front of the aircraft. The airframe is constructed of extruded polystyrene stock with carbon/glass fiber reinforcement at the leading edge. During collection of identification data, the UAV is piloted by remote control, and monitored through a telemetry link to a ground station.

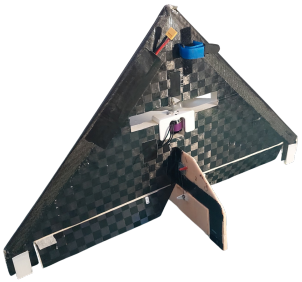


Figure 1: VTOL tail-sitter UAV.

3 SYSTEM MODELING

A standard approach is described with notations taken from [16]. In this paper, the roundness of the Earth is neglected, as well as its rotational speed; the ground-related coordinate system is inertial; the UAV is a rigid body.

3.1 Notations and Coordinate Systems

1. *Inertial Frame $\mathcal{F}^i$:* The i^i-axis is oriented toward north, the j^i-axis toward east, and the k^i-axis is directed toward the Earth's center.

2. *Body Frame $\mathcal{F}^b$:* The body frame is fixed to the aircraft and is used to describe its position and orientation. Its origin is located at the UAV's center of mass. Its axes (i^b, j^b, k^b) are oriented forward, to the right wing, and downward, respectively. The orientation of the body with respect to the inertial frame is described by the Euler angles (roll ϕ, pitch θ, yaw ψ), defined by the aerospace-standard ZYX rotation sequence.

3. *Wind Frame $\mathcal{F}^w$:* It is aligned with the direction of the airflow relative to the aircraft. Its origin is situated at the UAV's center of mass, and is defined by the axes (i^w, j^w, k^w). By definition, the relative airspeed vector lies along the i^w-axis.

To describe the equations of motion of the UAV, twelve state variables are used and described in Table 1.

Sym.	Description
p_n	Position along the i^i axis in $\mathcal{F}^i$
p_e	Position along the j^i axis in $\mathcal{F}^i$
p_d	Position along the k^i axis in $\mathcal{F}^i$
u	Velocity along the i^b axis in $\mathcal{F}^b$
v	Velocity along the j^b axis in $\mathcal{F}^b$
w	Velocity along the k^b axis in $\mathcal{F}^b$
ϕ	Roll angle
θ	Pitch angle
ψ	Yaw angle
p	Angular velocity about the i^b axis in $\mathcal{F}^b$
q	Angular velocity about the j^b axis in $\mathcal{F}^b$
r	Angular velocity about the k^b axis in $\mathcal{F}^b$

Table 1: State vector for the equations of motion

3.2 Flight Dynamics

Let $\boldsymbol{a}^b_{BE}$, $\boldsymbol{v}^b_{BE}$, $\boldsymbol{\omega}^b_{BE}$ respectively be the acceleration, the velocity and the angular velocity of the UAV with respect to the Earth in $\mathcal{F}^b$ such that:

$$\boldsymbol{a}^b_{BE} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}; \boldsymbol{v}^b_{BE} = \begin{pmatrix} u \\ v \\ w \end{pmatrix}; \boldsymbol{\omega}^b_{BE} = \begin{pmatrix} p \\ q \\ r \end{pmatrix}. \quad (1)$$

Based on $\boldsymbol{v}^b_{BE}$, the definitions of the airspeed V_a, the angle of attack α, and the sideslip angle β are given as:

$$V_a = \sqrt{u^2 + v^2 + w^2}; \quad (2)$$

$$\alpha = \tan^{-1}\left(\frac{w}{u}\right); \quad (3)$$

$$\beta = \sin^{-1}\left(\frac{v}{V_a}\right). \quad (4)$$

Moreover, the Newton-Euler laws of motion write:

$$\dot{\boldsymbol{v}}^b_{BE} = \frac{\boldsymbol{F}^b}{m} - \boldsymbol{\omega}^b_{BE} \times \boldsymbol{v}^b_{BE}; \quad (5)$$

$$\dot{\boldsymbol{\omega}}^b_{BE} = \boldsymbol{I}^{-1}(\boldsymbol{M}^b - \boldsymbol{\omega}^b_{BE} \times \boldsymbol{I}\boldsymbol{\omega}^b_{BE}), \quad (6)$$

http://www.imavs.org/

where $\boldsymbol{F}^b$ are the resultant forces, $\boldsymbol{M}^b$ the resultant moments in the body frame, and $\boldsymbol{I}$ the inertia matrix.

This set of equations defines the translational (5) and rotational dynamics (6), respectively. However, in the identification process, an alternative formulation of translational dynamics is preferred, valid under the assumption of no wind. It is obtained by isolating u, v, w in Equations (2–4), and get $\dot{u}$, $\dot{v}$, $\dot{w}$ after differentiation with respect to time. These components are substituted into Equation (5). The formulation is given as in [5]:

$$\dot{V}_a = \frac{1}{m}(\cos\alpha\cos\beta\, X + \sin\beta Y + \sin\alpha\cos\beta\, Z); \tag{7}$$

$$\dot{\alpha} = -\tan\beta\cos\alpha\, p + q - \tan\beta\sin\alpha\, r + \frac{1}{mV_a}\left(-\frac{\sin\alpha}{\cos\beta}X + \frac{\cos\alpha}{\cos\beta}Z\right); \tag{8}$$

$$\dot{\beta} = \sin\alpha\, p - \cos\alpha\, r + \frac{1}{mV_a}(-\cos\alpha\sin\beta\, X + \cos\beta\, Y - \sin\alpha\sin\beta\, Z), \tag{9}$$

where $(X, Y, Z)^T$ are the coordinates of the resultant force $\boldsymbol{F}^b$ along the body frame axes.

3.3 Actuators

1. *Servomotors:* They actuate the elevons and rudder of the UAV. Their dynamics are not negligible, hence, each is modeled by a first-order transfer function with time constant τ, along with a pure time delay named τ_a.

2. *Motor-Propeller Model:* The propulsion chain is composed by the ESC, the motor and the propeller. The ESC acts as a switching voltage stage that converts the throttle command δ_t into an effective motor input voltage u_m as:

$$u_m = V_{bat}\delta_t, \tag{10}$$

where V_{bat} is the battery voltage. The transfer from u_m to the motor angular speed Ω is nonlinear and modeled as a Hammerstein model: at the input, a polynomial function captures the static nonlinearity of the ESC-motor pair:

$$\Omega_{ss} = a_{esc}u_m^2 + b_{esc}u_m, \tag{11}$$

where Ω_{ss} is the steady-state motor speed. At the output, the dynamics are modeled as a second-order transfer function as:

$$\frac{\Omega}{\Omega_{ss}} = \frac{1}{1 + \frac{2\zeta}{\omega_0}s + \frac{1}{\omega_0^2}s^2}, \tag{12}$$

where ω_0 is the motor natural frequency, ζ its damping ratio and s is the Laplace variable.

3.4 Forces and Moments

1. *Gravitational Force:* In the inertial frame, it can be expressed by:

$$\boldsymbol{F}_g^i = \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix}, \tag{13}$$

where g is the acceleration due to gravity.

2. *Aerodynamic Forces and Moments:* Aerodynamic forces are expressed as follows in the wind frame:

$$\boldsymbol{F}_{aero}^w = -\frac{1}{2}\rho V_a^2 S \begin{pmatrix} C_D \\ C_Y \\ C_L \end{pmatrix}, \tag{14}$$

where ρ is the air density, S the reference surface, $C_{\{D,Y,L\}}$ the drag, side-force and lift coefficients, respectively. Aerodynamic moments are expressed as follows in the wind or body frame:

$$\boldsymbol{M}_{aero}^b = \boldsymbol{M}_{aero}^w = \frac{1}{2}\rho V_a^2 S \begin{pmatrix} bC_l \\ cC_m \\ bC_n \end{pmatrix}, \tag{15}$$

where b is the wingspan of the UAV, c its wing chord, $C_{\{l,m,n\}}$ the rolling, pitching and yawing moment coefficients, respectively.

The air density ρ is given by an implementation of the ISA atmosphere model, and computed from the altitude of the vehicle.

Then, the following decomposition of the aerodynamic coefficients is used, respectively for the longitudinal and lateral models:

$$\begin{pmatrix} C_D \\ C_L \\ C_m \end{pmatrix} = \begin{pmatrix} C_{D0} \\ C_{L0} \\ C_{m0} \end{pmatrix} + \begin{pmatrix} C_{D\alpha^2}\alpha & \frac{c}{2V_a}C_{Dq} \\ C_{L\alpha} & \frac{c}{2V_a}C_{Lq} \\ C_{m\alpha} & \frac{c}{2V_a}C_{mq} \end{pmatrix} \begin{pmatrix} \alpha \\ q \end{pmatrix} + \begin{pmatrix} C_{D\delta_e} \\ C_{L\delta_e} \\ C_{m\delta_e} \end{pmatrix} \delta_e; \tag{16}$$

$$\begin{pmatrix} C_Y \\ C_l \\ C_n \end{pmatrix} = \begin{pmatrix} 0 \\ C_{l0} \\ C_{n0} \end{pmatrix} + \begin{pmatrix} C_{Y\beta} & \frac{b}{2V_a}C_{Yp} & \frac{b}{2V_a}C_{Yr} \\ C_{l\beta} & \frac{b}{2V_a}C_{lp} & \frac{b}{2V_a}C_{lr} \\ C_{n\beta} & \frac{b}{2V_a}C_{np} & \frac{b}{2V_a}C_{nr} \end{pmatrix} \begin{pmatrix} \beta \\ p \\ r \end{pmatrix} + \begin{pmatrix} C_{Y\delta_a} & C_{Y\delta_r} \\ C_{l\delta_a} & C_{l\delta_r} \\ C_{n\delta_a} & C_{n\delta_r} \end{pmatrix} \begin{pmatrix} \delta_a \\ \delta_r \end{pmatrix}, \tag{17}$$

where $\delta_{\{a,e,r\}}$ are the roll, pitch and yaw control inputs, respectively.

3. *Propulsion Forces and Moments:* In the body frame, propulsion forces can be expressed using standard model from blade theory by:

$$\boldsymbol{F}^b_{prop} = \begin{pmatrix} \rho D_p^4 (\frac{\Omega}{2\pi})^2 C_T \\ 0 \\ 0 \end{pmatrix}, \tag{18}$$

where D_p is the diameter of the propeller. Besides, propulsion moments are defined as follows in the body frame:

$$\boldsymbol{M}^b_{prop} = \begin{pmatrix} \rho D_p^5 (\frac{\Omega}{2\pi})^2 C_Q \\ 0 \\ 0 \end{pmatrix}. \tag{19}$$

4. *Gyroscopic and Reaction Wheel Effect:* The gyroscopic moment due to rotation of the motor can be expressed as follows in the body frame:

$$\boldsymbol{M}^b_{gyro} = I_{pr} \begin{pmatrix} -\dot{\Omega} \\ -\Omega r \\ \Omega q \end{pmatrix}, \tag{20}$$

where I_{pr} is the moment of inertia of the rotor and the propeller.

4 NONLINEAR SYSTEM IDENTIFICATION

Based on the nonlinear model of the system derived in Section 3 and on data collected during flight tests, the overall goal of the method is to estimate the aerodynamic coefficients appearing in Equations (16) and (17). Previous tests have shown that some coefficients are very small and have a negligible effect on the outputs, in particular C_{L_q}, C_{n_0} and the coupling coefficients $C_{Y_{p,r}}$, C_{l_r}, C_{m_p} and C_{n_p}. Therefore, they are fixed to zero and will not be estimated in this paper. Besides, the available data are insufficient to estimate the second-order term $C_{D_{\alpha^2}}$, which can be obtained through CFD calculations.

4.1 Motor-Propeller Model Coefficients Estimation

The parameters of Equations (11–12) were determined beforehand using a dedicated test bench to measure the motor speed and supply voltage during testing. Parameters a_{ESC} and b_{ESC} were first identified from the steady-state function $\Omega_{ss}(u_m)$. Then, parameters ζ and ω_0 were estimated to fit step variations. The propeller coefficients C_T and C_Q were estimated from propeller performance curves given by the manufacturer [1] [2]. While the servomotors time constant τ was computed from measurements made on a dedicated test bench, the delay τ_a was estimated directly from input-output flight data, thus taking into account not only the servomotor delay but also additional delays in the loop. All the system parameters are listed in Table 2.

[1] https://m-selig.ae.illinois.edu/props/propDB.html
[2] https://www.apcprop.com/technical-information/performance-data

Sym.	Name	Value
m	Mass	$0.750\,\mathrm{kg}$
x_{CG}	Position of CG /nose	$-0.21\,\mathrm{m}$
I_{11}	Moment of inertia (x-axis)	$0.009\,09\,\mathrm{kg\,m^2}$
I_{22}	Moment of inertia (y-axis)	$0.007\,29\,\mathrm{kg\,m^2}$
I_{33}	Moment of inertia (z-axis)	$0.0157\,\mathrm{kg\,m^2}$
I_{13}	x-z product of inertia	$-0.000\,458\,\mathrm{kg\,m^2}$
b	Wingspan	$0.80\,\mathrm{m}$
c	Wing chord	$0.25\,\mathrm{m}$
S	Reference surface	$0.20\,\mathrm{m^2}$
V_{bat}	Battery voltage	$16.0\,\mathrm{V}$
τ	Servos time constant	$0.013\,\mathrm{s}$
τ_a	Servos actuator delay	$0.033\,\mathrm{s}$
a_{esc}	Quadratic ESC coef.	$-14.35\,\mathrm{rad\,s^{-1}\,V^{-2}}$
b_{esc}	Linear ESC coef.	$287.85\,\mathrm{rad\,s^{-1}\,V^{-1}}$
ω_0	Motor angular freq.	$17.94\,\mathrm{rad\,s^{-1}}$
ζ	Motor damping ratio	1.86
C_T	Thrust coef.	0.146
C_Q	Torque coef.	0.066

Table 2: Predetermined system parameters

4.2 Excitation Trajectory

Excitation signals are applied to each axis independently and are superimposed to pilot commands. By this mean, the pilot can correct the trajectory of the UAV if needed, in order to stay close to level-flight in straight line. Pilot's corrections are kept as small as possible. Several excitation signals could be used [6, 8]. In this work, they consist in 3-level step inputs, which are rather straightforward to implement.

Both amplitude and frequency of the multistep have been adjusted empirically during flight via the remote control. The amplitude has been tuned to ensure sufficient signal-to-noise ratio while keeping deviations from the nominal trajectory within acceptable limits. Likewise, the frequency has been adapted to the system dynamics so that quasi-steady-state conditions are reached after each excitation step.

Control input signals, as well as the measured angular velocities, are shown in Figure 2 for both roll and pitch axes. The yaw dynamics have not been considered in this study and no excitation is then provided on this axis.

Vectors of measured values are extracted from PX4 flight logs and uniformly resampled onto a common time base. Quaternions from the "vehicle-attitude" topic are used to compute the rotation matrix from the local to the body frame, as well as the Euler angles. Linear velocities and positions come from the "vehicle-local-position" topic, while angular rates come from "vehicle-angular-velocity". Accelerometer measurements (specific force) come from "vehicle-acceleration" and are used to compute body inertial accelerations by adding gravity. Measurements of V_a, α and β are computed from definitions (2–4). Control inputs are taken directly from the "torque-setpoint" and "thrust-setpoint" topics.

http://www.imavs.org/

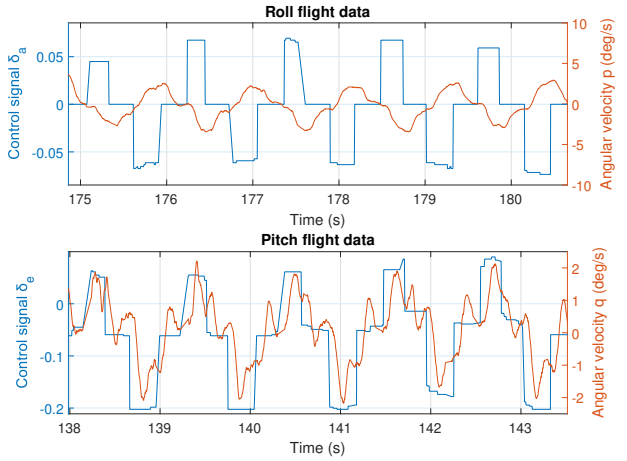


Figure 2: Roll and pitch angular velocities with control input signals.

4.3 *Aerodynamic Coefficient Estimation Method*

Experience shows that attempting to estimate the complete set of aerodynamic coefficient derivatives described in Equations (16) and (17) in a single optimization problem leads to convergence stalls or drift towards non-physical values (due to local minima, compensations between parameters, strong dependence on the initial guess). To reduce the complexity, the coefficients are estimated in seven stages, each stage involving estimation of a subset of one to four coefficients only, and the selection of a single output, as shown in Table 3. This approach eliminates strong parameter interdependence and yields robust convergence to physically meaningful values.

At each stage, the considered aerodynamic coefficients are estimated with the output-error method [5, 6]. The channel-wise accumulated squared errors in angular rate ($\boldsymbol{\epsilon}_\omega$) and acceleration ($\boldsymbol{\epsilon}_a$) are defined as:

$$\boldsymbol{\epsilon}_\omega(j) = \sum_i \left(\boldsymbol{\omega}^b_{BE,s}(i,j) - \boldsymbol{\omega}^b_{BE,m}(i,j)\right)^2; \qquad (21)$$

$$\boldsymbol{\epsilon}_a(j) = \sum_i \left(\boldsymbol{a}^b_{BE,s}(i,j) - \boldsymbol{a}^b_{BE,m}(i,j)\right)^2, \qquad (22)$$

where $i = 1, ..., N$ is the sample time index, $j = 1, 2, 3$ the body axes, and where "s", "m" indicate simulated and measured outputs, respectively. These notations will be kept in the next paragraphs. Vectors $\boldsymbol{\epsilon}_\omega$ and $\boldsymbol{\epsilon}_a$ are three-dimensional.

The scalar criterion J_k to be minimized by the optimization procedure is the weighted least-squares error computed as the squared differences between simulated and measured flight data as follows:

$$J_k = \boldsymbol{w}_k \begin{bmatrix} \boldsymbol{\epsilon}_\omega \\ \boldsymbol{\epsilon}_a \end{bmatrix}. \qquad (23)$$

The weight vector $\boldsymbol{w}_k$ allows selective activation or scaling of each channel, where $k = 1, ..., 7$ is the coefficient subset index (for example, $\boldsymbol{w}_k = [0, 1, 0, 0, 0, 1]$ includes only pitch rate and z-axis acceleration errors in the cost).

k	**Coefs.**	**Submodel eq.**	**Meas. inputs**	**Weight $\boldsymbol{w}_k$**
1	C_{L_0} C_{L_α} $C_{L_{\delta_e}}$	α dynamics from (8), (2)	$\delta_{\{a,e,r,t\}}$ p_m, q_m, r_m u_m, v_m, w_m	$[0,0,0,...$ $0,0,1]$
2	C_{D_0}	α dynamics from (8), (2)	$\delta_{\{a,e,r,t\}}$ p_m, q_m, r_m u_m, v_m, w_m	$[0,0,0,...$ $1,0,0]$
3	C_{Y_β} $C_{Y_{\delta_a}}$ $C_{Y_{\delta_r}}$	α, β from (8)-(9) and (2)	$\delta_{\{a,e,r,t\}}$ p_m, q_m, r_m u_m, v_m, w_m	$[0,0,0,...$ $0,1,0]$
4	C_{m_0} C_{m_α} $C_{m_{\delta_e}}$ C_{m_q}	q from (6) V_a, α from (7)-(8)	$\delta_{\{a,e,r,t\}}$ p_m, q_m, r_m	$[0,1,0,...$ $0,0,0]$
5	C_{l_0} C_{l_β} C_{l_p} $C_{l_{\delta_a}}$ $C_{l_{\delta_r}}$	p from (6) V_a, α, β from (7)-(9)	$\delta_{\{a,e,r,t\}}$ q_m, r_m	$[1,0,0,...$ $0,0,0]$
6	C_{D_0} C_{D_q} $C_{D_{\delta_e}}$	V_a, α from (7)-(8)	$\delta_{\{a,e,r,t\}}$ p_m, q_m, r_m	$[0,0,0,...$ $1,0,0]$
7	C_{n_β} C_{n_r} $C_{n_{\delta_a}}$	r from (6) V_a, α, β from (7)-(9)	$\delta_{\{a,e,r,t\}}$ p_m, q_m	$[0,0,1,...$ $0,0,0]$

Table 3: Aerodynamic coefficient estimation

Each line k of Table 3 defines an optimization problem: it specifies the subset of coefficients to be estimated (second column), the associated sub-model equations (third column), the measurements used as inputs (fourth column), and the weighting vector $\boldsymbol{w}_k$ employed in the cost function J_k (last column).

Linear and angular velocities can be obtained either directly from measurements or from the translational and rotational dynamics. Accordingly, depending on the subset of coefficients to be estimated (i.e. the stage k in Table 3), V_a, α, β are computed either from definitions (2–4) using the measured velocity $\boldsymbol{v}^b_{BE,m}$, or by solving the translational dynamics differential equations (7–9). Similarly, the angular velocity is obtained either from measurements $\boldsymbol{\omega}^b_{BE,m}$, or from rotational dynamics in Equation (6). The chosen set of equations is indicated in the third column of Table 3.

4.4 *Aerodynamic Coefficient Estimated Values*

To estimate the aerodynamic coefficients, the output-error method is applied using the Nelder-Mead simplex algorithm [17]. The coefficients obtained for the tail-sitter UAV with the identification approach are given in Table 4, compared to CFD values obtained from steady RANS simulations using

http://www.imavs.org/

the DLR TAU-Code [18]. Since no yaw inputs were used, $C_{Y_{\delta_r}}$, $C_{l_{\delta_r}}$ and $C_{n_{\delta_r}}$ are set to zero. Other $C_{n_\cdot}$ coefficients associated with the yaw axis are still estimated.

Coefs.	Estimated	CFD	Abs. err.	Rel. err.
C_{L_0}	0.005	0.026	0.021	80.8 %
C_{L_α}	1.186	3.449	2.263	65.6 %
$C_{L_{\delta_e}}$	0.350	/	/	/
C_{D_0}	0.125	0.021	0.104	495.2 %
C_{D_q}	0.000	/	/	/
$C_{D_{\delta_e}}$	0.001	/	/	/
C_{Y_β}	-0.006	-0.929	0.923	99.4 %
$C_{Y_{\delta_a}}$	0.005	/	/	/
C_{l_0}	-0.004	0.000	/	/
$C_{l\beta}$	-0.005	-0.003	0.002	66.7 %
C_{l_p}	-0.091	/	/	/
$C_{l\delta_a}$	-0.054	/	/	/
C_{m_0}	0.036	0.000	/	/
C_{m_α}	-0.193	-0.196	0.003	1.5 %
$C_{m_{\delta_e}}$	0.243	/	/	/
C_{m_q}	-3.427	/	/	/
C_{n_β}	0.100	0.169	0.069	40.8 %
C_{n_r}	-0.085	/	/	/
$C_{n_{\delta_a}}$	0.026	/	/	/

Table 4: Estimated aerodynamic coefficients compared to CFD results

The estimated values are overall consistent with the CFD predictions, and the discrepancies can be traced to modeling assumptions. First, the CFD simulations are performed on an ideal airfoil geometry: surface roughness, wing-mounted hardware (Pitot tube, protruding batteries, cables...) and, most importantly, the rotor disk located in the middle of the wing is not taken into account. Consequently, the lift-curve slope estimated from flight data, $C_{L_\alpha} = 1.186$ is significantly lower than the CFD value of 3.449. The same reasoning explains why the zero-lift drag coefficient exceeds the CFD result ($C_{D_0} = 0.125$ vs. 0.021 with the CFD); the real aircraft experiences more drag than the ideal laminar-flow profile assumed in CFD.

Coefficient C_{Y_β} also shows a large discrepancy of -0.006 vs. -0.929. This is due to the lack of yaw excitation data in the identification dataset, leading to low observability of the corresponding dynamics. By contrast, the static pitch-stability derivative $C_{m_\alpha} = -0.193$ matches the CFD value of -0.196 very closely and the values for the yaw-stability derivative C_{n_β} are close as well, confirming that the identification method captures these important stability terms with high fidelity. The non-zero value of C_{l_0} might traduce an asymmetry in the wing or elevons structure. Finally, it should be noted that the control-derivative coefficients (e.g. $C_{L_{\delta_e}}$ and $C_{m_{\delta_e}}$) and the dynamic coefficients (e.g. C_{m_q} and C_{l_p}) are notoriously difficult to obtain from CFD: the former require a large number of simulations for each control-surface deflection, while the latter demand complex coupling between CFD and dynamics. System identification, on the other hand, yields an estimate for these values.

The fitting results of the model to the flight data are shown in Figures 3 and 4. Two datasets collected during two different flights are used, one for identification and one for validation. The fitting percentage to evaluate the accuracy of the identification process is computed from a normalized root-mean-squared error, and given by:

$$f_{\%} = 100 \left(1 - \frac{\|\boldsymbol{y}_s - \boldsymbol{y}_m\|}{\|\boldsymbol{y}_m - \bar{\boldsymbol{y}}_m\|} \right), \tag{24}$$

where $\boldsymbol{y}_s$ is the simulated output vector, $\boldsymbol{y}_m$ the measured output vector, $\bar{\boldsymbol{y}}_m$ the sample mean of $\boldsymbol{y}_m$, and $\|\cdot\|$ denotes the Euclidean (2-)norm.

The fitting values for both identification and validation datasets are satisfactory. Indeed, on both datasets, the roll angular velocity p has a fit around $50\,\%$, and the acceleration a_y fit is higher than $80\,\%$. Moreover, pitch angular velocity q and acceleration a_z also have a fit between $40\,\%$ and $60\,\%$ on the identification dataset. On the validation dataset, the fits are lower for the latter axes, which can be explained by visible offsets. These can be attributed to differences in mass distribution between the tested configuration and the nominal model, as well as to trim changes commanded by the pilot via the remote control during flight. These effects modify the effective trim state and control biases seen in the measurements, but are not fully captured by the nominal aerodynamic model, which explains the observed offsets. Relatively low fit values on acceleration a_x is likely caused by the differing conditions between the motor-propeller test bench setup and real-world flight, and by the presence of wind during the experiments. Finally, the low yaw-rate fits (around 13%) are expected, because no yaw excitation have been used in this study.

Figures 3 and 4 also show α and β as predicted by the model, based on translational dynamics (8-9). On both datasets, β remains almost zero, which shows very low sideslip. In addition, α stays between $0°$ and $15°$, which is realistic in terms of angle of attack variations. As an additional validation, the model's predictions for the airspeed V_a agree well with the measurements on both identification (Figure 3) and validation (Figure 4) datasets, even though V_a was not used during the identification process.

5 CONCLUSION

This paper presents a generic nonlinear system identification method and its application to aerodynamic coefficient estimation of a tail-sitter VTOL drone in fixed-wing mode. The approach only needs a few dozen seconds of flight data, which avoids performing numerous measurement campaigns. The aerodynamic coefficients are estimated in stages using the output-error method, which eliminates strong parameter

http://www.imavs.org/

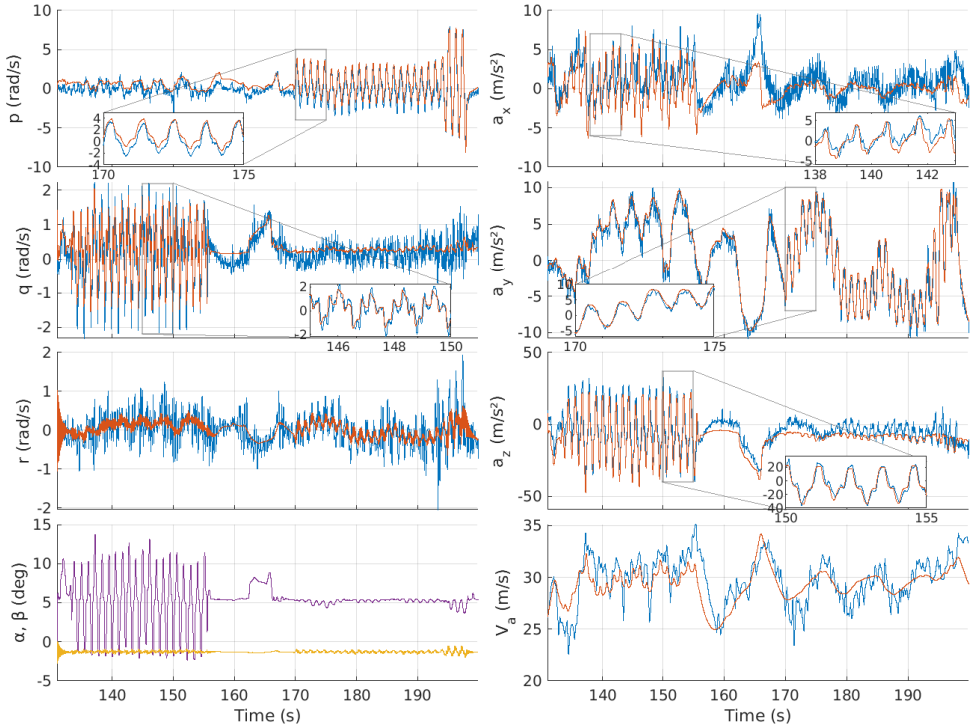


Figure 3: Identification results, with flight data in blue, model data in orange, α in purple and β in yellow. The fitting values are the following: $f_p = 46.5\%$, $f_q = 41.4\%$, $f_r = 13.3\%$, $f_{a_x} = 4.7\%$, $f_{a_y} = 82.4\%$, $f_{a_z} = 51.2\%$.

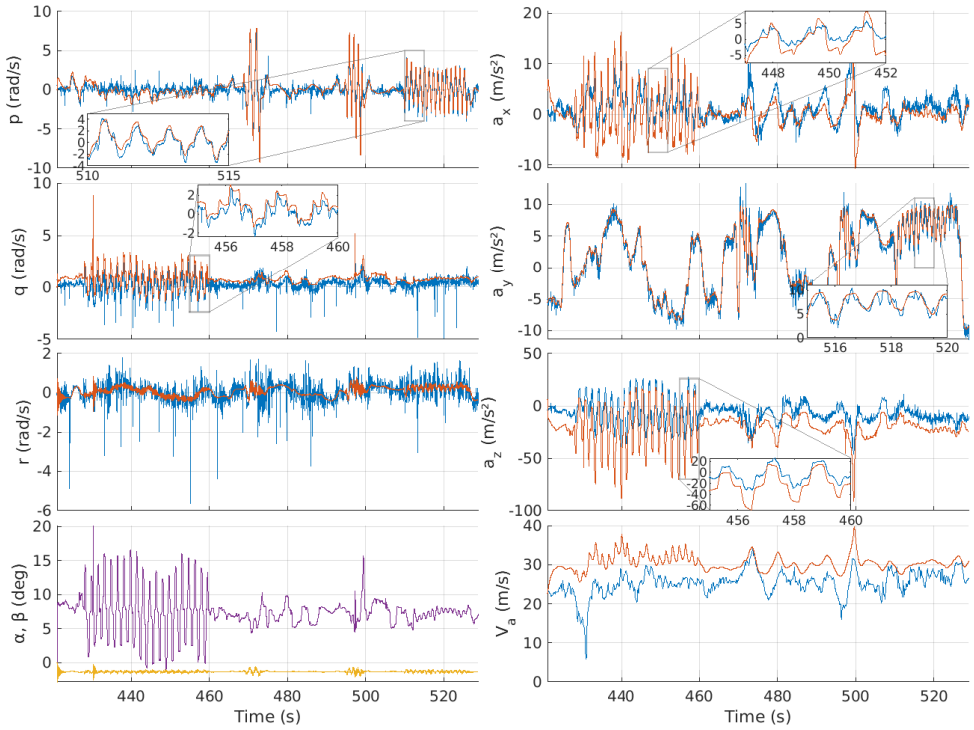


Figure 4: Validation results, with flight data in blue, model data in orange, α in purple and β in yellow. The fitting values are the following: $f_p = 54.1\%$, $f_q = -0.5\%$, $f_r = 13.6\%$, $f_{a_x} = -4.1\%$, $f_{a_y} = 84.5\%$, $f_{a_z} = -35.1\%$.

http://www.imavs.org/

interdependence and yields robust convergence to physically meaningful values. A limitation of the method is its sensitivity to data biases, due e.g. to miscalibrated accelerations and presence of wind during tests. Nevertheless, such biases mostly affect offset coefficients which have no impact on the linearized dynamics used for control design. Overall, the results are consistent with values predicted from CFD simulations, and satisfactory model fits to independent flight test data validate the entire methodology. Future work will include assessing the robustness of the methodology using simulated data. Besides, coefficients validity will be extended from fixed-wing flight to hover flight through the use of high-angle-of-attack analytical models.

REFERENCES

[1] Guillaume JJ Ducard and Mike Allenspach. Review of designs and flight control techniques of hybrid and convertible VTOL UAVs. *Aerospace Science and Technology*, 118:107035, 2021.

[2] Adnan S Saeed, Ahmad B Younes, Chenxiao Cai, and Guowei Cai. A survey of hybrid unmanned aerial vehicles. *Progress in Aerospace Sciences*, 98:91–105, 2018.

[3] Michael Selig. Modeling full-envelope aerodynamics of small UAVs in realtime. In *AIAA Atmospheric Flight Mechanics Conference*, page 7635, 2010.

[4] Ziqing Ma. *Design and Control of a Tilt-Rotor Tailsitter Aircraft*. PhD thesis, TU Delft, 2026.

[5] Eugene A Morelli and Vladislav Klein. *Aircraft System Identification: Theory and Practice*, volume 2. Sunflyte Enterprises Williamsburg, VA, 2016.

[6] Ravindra V Jategaonkar. *Flight Vehicle System Identification: A Time Domain Methodology*. American Institute of Aeronautics and Astronautics, 2006.

[7] Nathan V Hoffer, Calvin Coopmans, Austin M Jensen, and YangQuan Chen. A survey and categorization of small low-cost unmanned aerial vehicle system identification. *Journal of Intelligent & Robotic Systems*, 74(1):129–145, 2014.

[8] Benjamin M Simmons, James L Gresham, and Craig A Woolsey. Flight-test system identification techniques and applications for small, low-cost, fixed-wing aircraft. *Journal of Aircraft*, 60(5):1503–1521, 2023.

[9] Vishwas R Puttige and Sreenatha G Anavatti. Real-time system identification of unmanned aerial vehicles: A multi-network approach. *Journal of Computers*, 3(7):31–38, 2008.

[10] Andrei Dorobantu, Austin Murch, Bérénice Mettler, and Gary Balas. System identification for small, low-cost, fixed-wing unmanned aircraft. *Journal of Aircraft*, 50(4):1117–1130, 2013.

[11] Nirmal Kumar, Subrahmanyam Saderla, and Y Kim. Parametric model identification of delta wing UAVs using filter error method augmented with particle swarm optimisation. *The Aeronautical Journal*, 127(1312):982–1008, 2023.

[12] Weifeng Zhou, Shengyang Chen, Ching-Wei Chang, Chih-Yung Wen, Chih-Keng Chen, and Boyang Li. System identification and control for a tail-sitter unmanned aerial vehicle in the cruise flight. *IEEE Access : Practical Innovations, Open Solutions*, 8:218348–218359, 2020.

[13] Hosun Lee, Jayden D Lee, and Hyochoong Bang. Aerodynamic model identification of a VTOL tailsitter UAV using sparse identification of nonlinear dynamics. In *ICAS Proceedings*, 2024.

[14] Carl Olsson, Sebastian L Verling, Thomas Stastny, and Roland Siegwart. Full envelope system identification of a VTOL tailsitter UAV. In *AIAA Scitech 2021 Forum*, page 1054, 2021.

[15] Lorenz Meier, Dominik Honegger, and Marc Pollefeys. PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *IEEE International Conference on Robotics and Automation*, pages 6235–6240, 2015.

[16] Randal W Beard and Timothy W McLain. *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012.

[17] Jeffrey C Lagarias, James A Reeds, Margaret H Wright, and Paul E Wright. Convergence properties of the Nelder–Mead simplex method in low dimensions. *SIAM Journal on Optimization*, 9(1):112–147, 1998.

[18] Dieter Schwamborn, Thomas Gerhold, and Ralf Heinrich. The DLR TAU-code: Recent applications in research and industry. In *ECCOMAS CFD: Proceedings of the European Conference on Computational Fluid Dynamics*, Egmond aan Zee, the Netherlands, 2006.

http://www.imavs.org/

TriMorph: The Transformative Tricopter

J. Lee†, J-M Ingargiola‡, and S. Viollet‡*
†Department of Mechanical Engineering, Massachusetts Institute of Technology, MA, USA
‡CNRS / Aix-Marseille Université, ISM, Marseille, France

http://www.imavs.org/

Abstract

Bio-inspired robotic systems increasingly exploit morphological adaptation to operate under varying environmental constraints. While morphing mechanisms have been investigated for unmanned aerial vehicles (UAVs), relatively little attention has been given to tricopters, despite their favorable efficiency and agility characteristics. This paper presents TriMorph, a transforming tricopter capable of continuously varying its frame through a single-actuator truss mechanism. A geometric model relating actuator motion to the deploy angle is derived and incorporated into a configuration-dependent dynamic formulation. Experimental flights demonstrate consistent behavior across configurations, highlighting morphology as a control input that modifies the inertia matrix. These results suggest a pathway toward aerial robots that adapt their physical structure to changing task requirements. The prototype increases its top-down area by more than $3\times$ and principal inertias by up to $2.9\times$. These results demonstrate that rotational geometric morphing directly modulates the inertia matrix in flight, thereby expanding the operational envelope of tricopter UAVs.

1 Introduction

Multirotor UAVs are increasingly deployed in logistics, disaster response, agriculture, and defense applications. However, most platforms rely on fixed-geometry airframes that cannot adapt their morphology to changing environmental or task constraints. In contrast, biological flyers, such as birds, dynamically adjust wing morphology to balance agility, stability, and energy efficiency across varying conditions. Not only are these aerial creatures able to vary their aerodynamics at will with their foldable wings and overlapping feathers, they are able to perform rapid adaptation and optimization suitable for hunting, fleeing and energy efficiency [1]. By integrating characteristics of wildlife, drones would be able to navigate through narrow openings needed for search and rescue, rapidly adapt to atmospheric conditions, optimize energy consumption, and minimize radio wave reflections for military stealth, to name a few examples. Although there exists research on this topic with transforming drones [2], [3], [4], tricopters, unlike other multirotor designs, offer speed and agility ([5]). TriMorph takes the leap in bringing our rigid aerial vehicles one step closer to the functionalities of flying animals, while retaining the strength and robust qualities of a drone. The main contributions of this work are the design of a morphing tricopter architecture, dynamic modeling that captures configuration-dependent behavior, and a control implementation enabling flight.



Figure 1: TriMorph expansion evolution.

2 Related Work

Although similar shape-shifting drones have been developed [2], [4], [6], [7], [8], [9], [10] the degree of their transformations is often limited to one arm actuated movements, limiting their expansion and therefore, range of adaptation. For example, in [2], the fully expanded and contracted states of the 'Foldable Drone' are targeted for object inspection and crevice navigation and are not suitable for extreme changes in environmental conditions, where rapid change in drone dynamics is necessary while maintaining stability. On the other hand, Morpho-Copter [4] is only able to change the angle of its arms, and its four servo motors make it prone to failures, according to the authors. As for the sliding arm quadcopter [7], it changes its shape through arm extensions, consequently moving its center of mass; however, this design also does not change its rotational inertia significantly, as seen from its minimal pitch and yaw rotations.

Given the complexity of current morphing drones, with numerous servo motors, the drones do not achieve large increases in width, which is why TriMorph utilizes truss-like mechanisms to simplify the servo motor complexity while maximizing adaptation and stability with expansion [11]. Figure 1 illustrates the expansion evolution of TriMorph from

*Email address: stephane.viollet@cnrs.fr

its compact configuration to its fully deployed state, highlighting the continuous change in planform area span.

To contextualize the performance indices of TriMorph against prominent platforms in the literature, a comparative overview is provided in Table 1.

Table 1: Comparative analysis of TriMorph against state-of-the-art morphing UAVs.

Platform	Type	Morphing Actuators	Expansion Ratio (Area / Span)	Inertia Modulation Factor (I_{zz})	CoG Shift During Flight
Foldable Drone [2]	Quadrotor	4 Servos	$\sim 1.2\times$ (Area)	Low	Negligible
Morpho-Copter [4]	Quad-Bi	4 Servos	$\sim 1.2\times$ (Area)	Low	Negligible
Sliding Arm Quad [7]	Quadrotor	2 Servos	$\sim 1.2\times$ (Span)	Minimal (xy only)	Pronounced
TriMorph	**Tricopter**	**1 Servo**	$> 3.2\times$ **(Area)**	**Up to** $2.9\times$	**Negligible** (< 0.8 **mm**)

Figure 1 illustrates the expansion evolution of TriMorph from its compact configuration to its fully deployed state, highlighting the continuous change in planform area span.

3 MECHANICAL DESIGN

3.1 Truss-Based Morphing Architecture

Many morphing UAVs rely on multiple actuators and rigid telescoping arms, increasing mass and mechanical complexity [11]. In contrast, TriMorph adopts a planar truss-inspired mechanism in which each arm behaves primarily as a two-force member. This allows structural loads to be carried predominantly in axial tension or compression rather than in bending, supporting lightweight construction while maintaining stiffness.

All twelve joints were manufactured in carbon-fiber reinforced polymer (PET-CF) using additive manufacturing. The truss layout ensures that the geometric expansion modifies the motor to centroid distance without introducing large off-axis bending loads in the members.

3.2 Joint Geometry and Stress Distribution

Arm connections experience the highest stresses in the structure because they transfer load between members while undergoing repeated rotation during morphing. Early prototypes exhibited fracture at the arm roots due to localized bending stresses. To mitigate this, the joints were redesigned with enlarged circular ends, increasing the load transfer area and reducing stress concentration. The peak stress at a geometric discontinuity is governed by:

$$\sigma_{\max} = K_t \sigma_{\text{nom}} \tag{1}$$

where K_t is the stress concentration factor. By replacing narrow rectangular ends with circular profiles, K_t is reduced, distributing load more uniformly into the arm body.

Additionally, the pin connection introduces bearing stress:

$$\sigma_b = \frac{F}{t d_p} \tag{2}$$

where t is the material thickness and d_p the pin diameter. Increasing the joint diameter increases the projected bearing area, lowering local compressive stresses and reducing crack initiation in the printed composite layers. This is particularly important for layer-wise printed polymers, which are susceptible to delamination under localized transverse loading. As a result, the rounded joint geometry significantly improved durability and prevented brittle fracture during repeated morphing cycles.

3.3 Primary Arm Bending Stiffness

The arm connecting the rotating frame to the motor experiences a different loading than the remaining members. Although most truss elements primarily carry axial forces, the motor arm acts as a cantilever beam that supports rotor thrust, gyroscopic torques, and inertial loads during maneuvering. Consequently, bending stress dominates its structural loading. To increase stiffness without significantly increasing mass, the motor arm was printed with twice the vertical height of the other arms. The bending stress in a beam is given by:

$$\sigma = \frac{Mc}{I} \tag{3}$$

where M is the bending moment, c is the distance to the outer fiber, and I is the second moment of area. For a rectangular cross-section:

$$I = \frac{bh^3}{12} \tag{4}$$

Thus, doubling the motor arm height increases I by a factor of $2^3 = 8$, and c by a factor of 2, reducing bending stress and deflection by factors of 4 and 8 respectively for the same loading. This modification significantly improved structural rigidity and reduced oscillations at the motor mount during flight, while adding minimal mass compared to uniformly thickening all members.

3.4 Low-Friction Rotational Interfaces

Rotational joints use $2\,\text{mm}$ PTFE spacers to reduce mass and avoid stick–slip during repeated morphing. PTFE exhibits very low surface energy and shear strength, producing a friction coefficient of approximately $\mu \approx 0.04$–0.10 against the carbon-fiber composite. The material also conforms slightly to surface irregularities, distributing contact pressure and limiting abrasive wear at the joint interface. This allowed smooth expansion and contraction without mechanical binding or observable wear during testing.

3.5 Actuation Strategy

Instead of a conventional tail tilt mechanism, TriMorph employs a coaxial motor unit in which counter-rotating propellers balance the net reaction torque generated by the propellers. A single servo rotates the top triangular plate, driving the truss mechanism and changing the radial motor distance.

The expansion angle α directly modifies TriMorph's frame and inertia while maintaining symmetry about the body z-axis. Because angular acceleration scales as:

http://www.imavs.org/

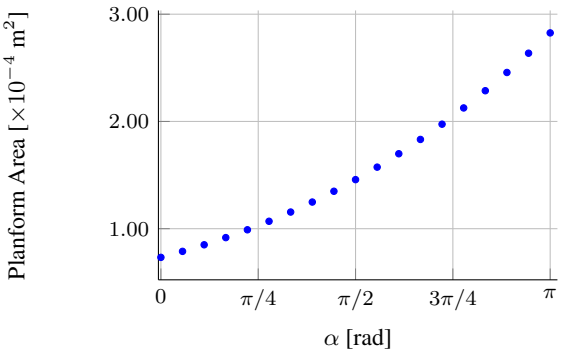


Figure 2: Planform Area *vs.* Deploy Angle (α), proportional to kx^2, where $k \in \mathbb{R}$.

$$\dot{\omega} = J^{-1}\tau \tag{5}$$

increasing the moment of inertia reduces angular acceleration for a given torque, improving disturbance resilience, while the compact configuration enables greater agility.

Through serial radio communication, we implemented a toggle that allows the drone to expand and contract mid-air. At an initial configuration, its planform area, defined by the equilateral triangle formed by the three motor hubs vertex-to-vertex, is 7,317.352 mm^2. TriMorph is capable of expanding to more than triple its original area to 23,482.812 mm^2. This relationship is depicted in Figure 2, where the planform area is plotted as a function of the deploy angle. For the principal-axis rotational inertia, the tricopter starts at $L_{xx} = 7.1 \times 10^{-4}\,\mathrm{kg\,m^2}$, $L_{yy} = 7.6 \times 10^{-4}\,\mathrm{kg\,m^2}$, $L_{zz} = 9.9 \times 10^{-4}\,\mathrm{kg\,m^2}$, and morphs to $L_{xx} = 1.7 \times 10^{-3}\,\mathrm{kg\,m^2}$, $L_{yy} = 1.6 \times 10^{-3}\,\mathrm{kg\,m^2}$, $L_{zz} = 2.9 \times 10^{-3}\,\mathrm{kg\,m^2}$.

This not only creates significant advantages in resilience to disturbances, but also enables the UAV to alter its dynamics midair in response to environmental conditions, such as air drag. Because the morphing mechanism relies entirely on a closed-loop internal truss structure driven by a single internal servo, the total mass of the vehicle remains strictly constant ($m_t = 345$ g) during adaptation. Furthermore, due to the rotational symmetry of the three-arm truss layout about the body z_b-axis, the theoretical Center of Gravity (CoG) experiences zero translation in the xy-plane ($\Delta x_G = 0, \Delta y_G = 0$). Experimentally, minor manufacturing and assembly tolerances result in a negligible CoG translation of less than 0.8 mm, which is easily rejected by the attitude controller's integral action. Finally, a custom triangular PCB board (Figure 4) is mounted on the drone to provide easily accessible integration of electronics.

4 Geometric Analysis of the Mechanism

We describe here the motion of one motor with respect to a rotation of the top triangular plate, hereafter referred to as the drive plate, around the vertical axis z_b by a known angle α. As shown in Figure 3, we consider the body frame $\mathcal{B} = \{O; \vec{x}_b, \vec{y}_b, \vec{z}_b\}$ to be fixed to the bottom triangular base plate. Points N and M move along a circle centered on point O and C, respectively. As such, distances ON, NM and CM are constant and named R_1, L and R_2, respectively. Similarly, point M is seen to be at the intersection of the circle centered at N with a radius L and the circle centered at C.

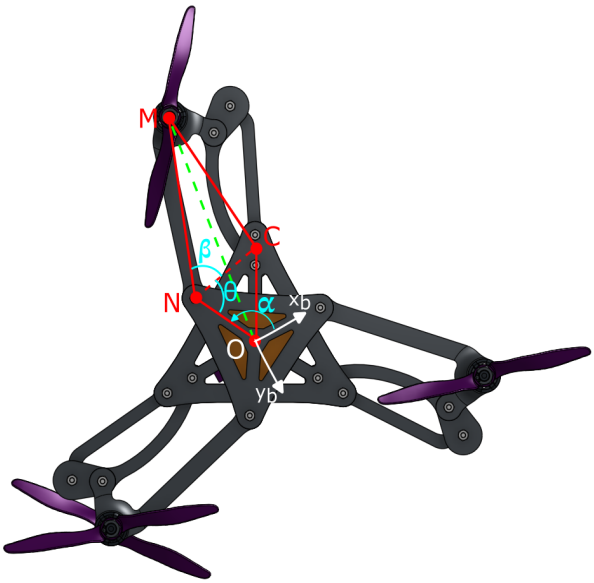


Figure 3: Diagram showing the motor's position vector dependent on drive plate rotation α and subsequent angles β and θ. The dotted green line represents the distance d. The body frame is considered attached to the bottom triangular base plate.

By definition, we have:

$$\overrightarrow{ON} = \begin{bmatrix} x_N \\ y_N \end{bmatrix} = R_1 \begin{bmatrix} \cos\alpha \\ \sin\alpha \end{bmatrix} \tag{6}$$

Let NC be the distance between the centers: $NC = \sqrt{(x_N - x_C)^2 + (y_N - y_C)^2}$, with x_c and y_c known by construction. In the triangle $\triangle NCM$, we know all three sides: L, R_2, and NC. Given this, we can find the internal angle at N (named β) using the law of cosines:

$$\cos(\beta) = \frac{NC^2 + L^2 - R_2^2}{2 \cdot NC \cdot L} \tag{7}$$

Then, the coordinates of M are simply a rotation and translation from point N. The angle of the vector $\overrightarrow{NC}$ relative to the x-axis is: $\theta = \mathrm{atan2}(y_N - y_C, x_N - x_C)$. Finally, we have the coordinates of M:

$$x_M = x_N + L\cos(\theta + \beta) \tag{8}$$

$$y_M = y_N + L\sin(\theta + \beta) \tag{9}$$

http://www.imavs.org/

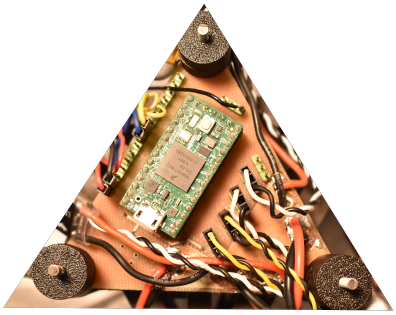


Figure 4: Custom-made triangular PCB board supporting the Teensy 4.0 board running the dRehmFlight autopilot [12].

The distance d_i (norm of $\overrightarrow{OM_i}$) between the motor M_i and O is $\sqrt{x_{Mi}^2 + y_{Mi}^2}$ and the orientation of $\overrightarrow{OM_i}$ is named $\theta_{M_i} = \text{atan2}(y_{Mi}, x_{Mi})$ with $i \in [1, 3]$. This distance, modeled by a four-bar (crank-rocker) mechanism, can be approximated as a single-bar mechanism. We measured an error as small as 2.2% between the real and theoretical distances d for three α angles ($0°$, $90°$ and $180°$). Therefore, we treat R_2 as constant in the simplified model. The theoretical geometry was simulated with Geogebra software.

5 DYNAMICAL MODEL

In this section, we develop the equations of motion (EOMs) for TriMorph using the classical Newton-Euler framework. Because the tricopter undergoes geometric morphing, the distance d_i between each motor M_i and the body center varies with the drive plate angle α. Figure 5 illustrates the compact configuration at $\alpha = 0°$, corresponding to the minimum motor-to-center distance d_i. Figure 6 shows the fully deployed configuration at $\alpha = 180°$, corresponding to the maximum radial extension of the motors and increased rotational inertia.

5.1 Forces and Torques

The forces and drag torques produced by each rotor are assumed to be proportional to the square of the angular speeds ω_i: $f_i = k_t\omega_i^2$ and $\tau_i = (-1)^i k_d\omega_i^2 \quad \forall i \in \{1, 2, 3, 4\}$. The force produced by the i^{th} rotor is thus:

$$\boldsymbol{f}_i = \begin{bmatrix} 0 \\ 0 \\ -k_t\omega_i^2 \end{bmatrix} \text{ and } \mathbf{F}_m^b = \begin{bmatrix} 0 \\ 0 \\ -k_t\left(\omega_1^2 + \omega_2^2 + \omega_{3,4}^2\right) \end{bmatrix} \tag{10}$$

Motors 3 and 4 are aligned, so the thrust is distributed fifty-fifty between the two motors. Because the rotors maintain a strictly perpendicular thrust orientation relative to the horizontal base plate throughout the morphing process, the total static thrust required to maintain a steady hover remains uniform across the full expansion range. Further experiments will be achieved to measure changes in power consumption when transitioning from the compact ($\alpha = 0°$) to the fully deployed state ($\alpha = 180°$), primarily caused by minor variations in downstream airflow interaction with the expanded structural frame.

Let (M_1, M_2, M_3) be the application points of the forces (f_1, f_2, f_3) respectively. Then the torque generated by the rotors with respect to the center of mass G can be expressed in the body frame as:

$$\mathbf{T}_r^b = \sum_{i=1}^{3} \boldsymbol{\tau}_{m,i}^b = \sum_{i=1}^{3} (\mathbf{G}_{Mi} \times \boldsymbol{f}_i) \tag{11}$$

where G_{Mi} is the vector of the distance of the i^{th} rotor from the center of gravity G defined as follows (Figure 3):

$$\mathbf{G} = \begin{bmatrix} | & | & | \\ \mathbf{G}_{M_1} & \mathbf{G}_{M_2} & \mathbf{G}_{M_3} \\ | & | & | \end{bmatrix} = \begin{bmatrix} d\cos\theta_{M1} & d\cos\theta_{M2} & -d\cos\theta_{M3} \\ d\sin\theta_{M1} & -d\sin\theta_{M2} & d\sin\theta_{M3} \\ 0 & 0 & 0 \end{bmatrix} \tag{12}$$

Then, we can apply equations 12 to 11 to express the torque produced by the rotors as:

$$\mathbf{T}_r^b = \begin{bmatrix} dk_t\left(-\omega_1^2\sin(\theta_{M1}) + \omega_2^2\sin(\theta_{M2}) - \omega_{3,4}^2\sin(\theta_{M3})\right) \\ dk_t\left(\omega_1^2\cos(\theta_{M1}) + \omega_2^2\cos(\theta_{M2}) - \omega_{3,4}^2\cos(\theta_{M3})\right) \\ 0 \end{bmatrix} \tag{13}$$

where we consider $d_1 = d_2 = d_3 = d$ and $(\omega_3^2 + \omega_4^2) = \omega_{3,4}^2$.

In addition to thrust-induced reaction torques, each rotor generates an aerodynamic drag torque about the body z-axis, opposing its direction of rotation. The total drag torque from the four rotors is therefore given by:

$$\mathbf{T}_d^b = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -k_d\left(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2\right) \end{bmatrix} \tag{14}$$

By grouping equations 10, 13 and 14, the expression for the total forces and torques described by the tricopter control mixer is obtained by:

$$\begin{bmatrix} F_x \\ F_y \\ F_z \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} \mathbf{F}_m^b \\ \mathbf{T}_r^b + \mathbf{T}_d^b \end{bmatrix} \tag{15}$$

5.2 Rigid Body Model

TriMorph is modeled as a rigid body of mass m_t. The propulsive force $\mathbf{F}_m^b$ is expressed in the body frame and transformed into the inertial (earth) frame for translational dynamics.

http://www.imavs.org/

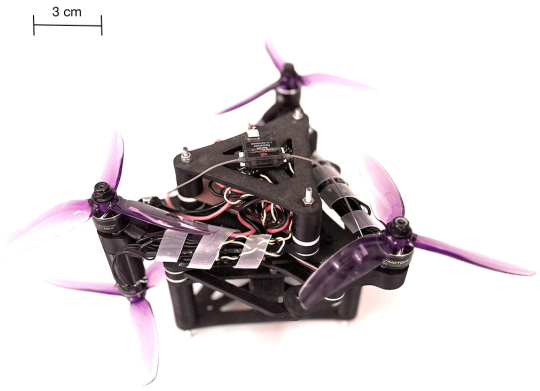


Figure 5: TriMorph in its most compact state at a drive plate rotation of 0°.

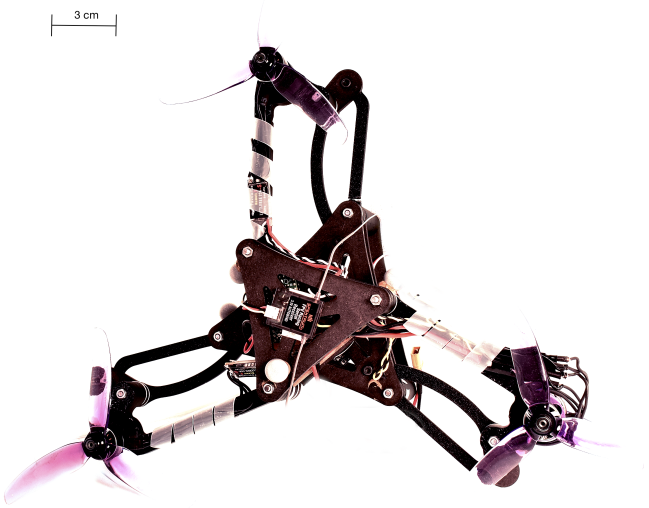


Figure 6: TriMorph in its fully deployed state at a drive plate rotation of 180°.

The gravitational force acting on the vehicle is given in the inertial frame as:

$$\mathbf{F}_g^e = \begin{bmatrix} 0 \\ 0 \\ m_t g \end{bmatrix}. \tag{16}$$

The translational dynamics are subjected to a configuration-dependent bluff-body aerodynamic drag force, where the effective drag profile scales proportionally with the planform area expansion shown in Figure 2. This might yield a highly streamlined, high-speed profile in the compact configuration and a high-drag, high-stability profile when fully expanded. This point remains to be shown in further experiments involving a windshaper.

We define $\boldsymbol{\xi}^e \triangleq \begin{bmatrix} x & y & z \end{bmatrix}^\top$, $\boldsymbol{\eta}^b \triangleq \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^\top$ and $\boldsymbol{\omega}^b \triangleq \begin{bmatrix} \omega_p & \omega_q & \omega_r \end{bmatrix}^\top$. Using the Newton-Euler framework, the translational dynamics of the tricopter is:

$$\ddot{\boldsymbol{\xi}}^e = \frac{1}{m_t}\left[\boldsymbol{R}_e^b(\boldsymbol{\eta})^\top \mathbf{F}_m^b + \mathbf{F}_g^e\right] \tag{17}$$

$$\dot{\boldsymbol{\eta}} = \boldsymbol{\Gamma}(\boldsymbol{\eta})^{-1}\boldsymbol{\omega}^b \tag{18}$$

$$\dot{\boldsymbol{\omega}}^b = \mathbf{J}^{-1}\left[\left(-\boldsymbol{\omega}^b \times \mathbf{J}\boldsymbol{\omega}^b\right) + \mathbf{T}_m^b\right] \tag{19}$$

with $\mathbf{J}$ the inertia matrix, the rotation matrix $\boldsymbol{R}_e^b = \boldsymbol{R}_x(\phi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_z(\psi)$ and $\boldsymbol{\Gamma}(\boldsymbol{\eta})$ given by:

$$\boldsymbol{\Gamma}(\boldsymbol{\eta}) = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\theta & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \tag{20}$$

6 CONTROL IMPLEMENTATION

The flight control system is based on the open-source dRehmFlight VTOL architecture [12]. The controller runs on a Teensy 4.x microcontroller (Figure 4) at loop frequencies exceeding 2 kHz, enabling low-latency attitude stabilization. For the tricopter configuration, a custom control mixer was developed to accommodate asymmetric thrust geometry and decouple yaw dynamics from roll and pitch. The pitch and roll PID outputs are distributed across the four primary lifting motors, while yaw control is handled independently to mitigate non-linear coupling effects introduced by the morphology. As shown in Figure 7, the control pipeline consists of state estimation, command mapping, attitude control and morphology-aware mixing.

6.1 *State Estimation*

Attitude is estimated using an IMU-based quaternion filter. Gyroscope integration provides short-term orientation propagation while accelerometer (gravity) and magnetometer (heading) measurements provide drift correction. The estimator follows the standard form:

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes \boldsymbol{\omega} - \beta\nabla J \tag{21}$$

where $\mathbf{q}$ is the attitude quaternion and β balances drift rejection and noise sensitivity. Euler angles (ϕ, θ, ψ) are derived from $\mathbf{q}$ for feedback control.

6.2 *Command Mapping*

User commands are mapped to normalized thrust and attitude setpoints:

$$u_T \in [0, 1] \tag{22}$$

$$\phi_d \in [-\phi_{\max}, \phi_{\max}] \tag{23}$$

$$\theta_d \in [-\theta_{\max}, \theta_{\max}] \tag{24}$$

$$r_d \in [-r_{\max}, r_{\max}] \tag{25}$$

http://www.imavs.org/

http://www.imavs.org/

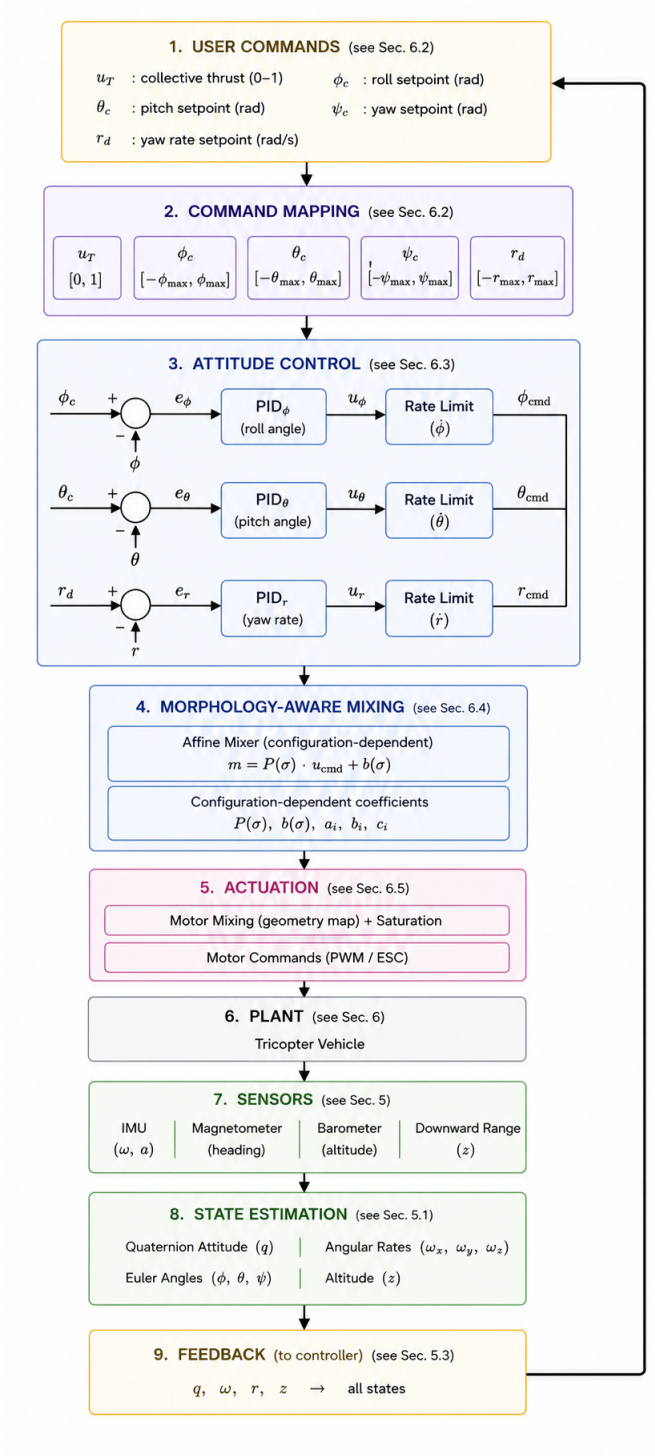


Figure 7: General control architecture of TriMorph's flight control system, showing command mapping, attitude control, configuration-dependent motor mixing, actuation, sensor feedback, and state estimation within each control cycle.

6.3 *Attitude Control*

Roll and pitch are regulated using PID control on angle error, with rate damping from gyroscope measurements:

$$e_\phi = \phi_d - \phi, \tag{26}$$

$$u_\phi = k_{p,\phi} e_\phi + k_{i,\phi} \int e_\phi dt - k_{d,\phi} \omega_x \tag{27}$$

Yaw control follows the same structure using yaw rate feedback and the integrator limiting prevents windup near zero thrust.

6.4 *Morphology-Aware Mixing*

Because the arm length d varies during morphing, the effectiveness of each rotor changes with configuration. Instead of retuning gains, we compensate using an affine mixer mapping control inputs to motor commands:

$$m_i = u_T + a_i u_\theta + b_i u_\phi + c_i u_r \tag{28}$$

where coefficients (a_i, b_i, c_i) encode configuration-dependent control authority. This allows stable flight across compact and deployed geometries without altering the high-level controller. With this flight controller, we are able to establish resilience in roll, pitch, and yaw, as the drone lifts off into hover state for the fully deployed configuration (see Figure 8).

7 Conclusion

In conclusion, integrating a morphing mechanism into a tricopter platform significantly enhances operational adaptability. By dynamically adjusting TriMorph's characteristic span, the vehicle transitions between a compact configuration optimized for maneuverability in confined environments and an extended configuration that improves attitude resilience and endurance during forward flight. This variable geometry enables real-time modulation of the system's moment of inertia and control authority, expanding the flight envelope beyond that of a fixed-geometry platform. When combined with the dRehmFlight VTOL control architecture, the morphing mechanism supports stable hovering flight within a single airframe. With this flight controller, we are able to implement a stable control in roll, pitch, and yaw, as the drone lifts off in its fully deployed state. Stability in intermediate states remains to be shown.

While preliminary gust rejection experiments successfully verified the immediate stability benefits of physical inertia modulation, future work will focus on extensive windshape characterization to accurately map dynamic transitions under continuous crosswind aerodynamic loading by means of a windshaper available at the lab.

Acknowledgment

We acknowledge the support of the Centre National de la Recherche Scientifique (CNRS), Aix-Marseille University and the Massachusetts Institute of Technology through

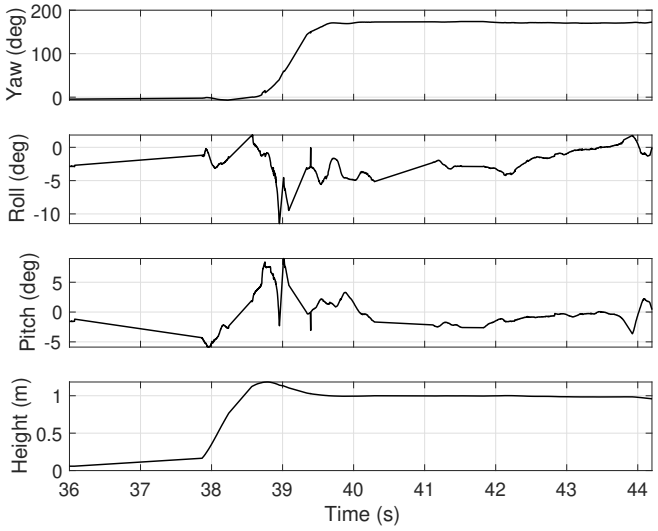


Figure 8: Experimental measurements of pitch, roll, yaw, and altitude over time before and after takeoff. The yaw response remains stable for several seconds following liftoff.

the MIT France MISTI Program. This research work was also supported by the French National Agency for Research (ANR) under the ResiRob project (ANR-24-CE33-4511-01).

REFERENCES

[1] S. Mintchev and D. Floreano. Adaptive morphology: A design principle for multimodal and multifunctional robots. *IEEE Robotics & Automation Magazine*, 23(3):42–54, September 2016.

[2] D. Falanga, K. Kleber, S. Mintchev, D. Floreano, and D. Scaramuzza. The foldable drone: A morphing quadrotor that can squeeze and fly. *IEEE Robotics and Automation Letters*, 4(2):209–216, April 2019.

[3] N. Bucki and M. W. Mueller. Design and control of a passively morphing quadcopter. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 9116–9122, 2019.

[4] H. Modi, H. Su, X. Liang, and M. Zheng. Morphocopter: Design, modeling, and control of a new transformable quad-bi copter. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, June 2025.

[5] Desh Deepak Sharma, Jeremy Lin, and Ayush Singh. Disturbance observer-based adaptive sliding mode control for variable dihedral and dual-mode yaw vectoring medium-size tricopter uav. *Scientific Reports*, 2026.

[6] Ju-Hee Lee, Sumin Jung, Jinwoo Jang, Yeji Han, Minchae Kang, Jihun Seong, Suyeon Seo, Eunsol Park, Haesol Kwak, and Min-Woo Han. Design challenges to expand the functionality of drones: Deformable rotorcraft and nature-inspired flapping drones. *Advanced Intelligent Systems*, page e202500790, 2025.

[7] R. Kumar, A. Deshpande, J. Wells, and M. Kumar. Flight control of sliding arm quadcopter with dynamic structural parameters. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2020.

[8] P. Pounds, R. Mahony, J. Gresham, P. Corke, and J. M. Roberts. Towards dynamically-favourable quad-rotor aerial robots. In *Proceedings of the Australasian Conference on Robotics and Automation*, 2004.

[9] V. Riviere, A. Manecy, and S. Viollet. Agile robotic fliers: A morphing-based approach. *Soft Robotics*, 2018.

[10] J. C. Gomez and E. Garcia. Morphing unmanned aerial vehicles. *Smart Materials and Structures*, 20(10):103001, 2011.

[11] David St-Onge and Clément Gosselin. Synthesis and design of a one degree-of-freedom planar deployable mechanism with a large expansion ratio. *Journal of Mechanisms and Robotics*, 8(2):021025, 2016.

[12] Nick Rehm. *dRehmFlight VTOL: Open-Source Flight Control and Stabilization*, 2020. Accessed February 2026.

http://www.imavs.org/

Weitflug: A Research-Oriented Fork of Betaflight for Agile Autonomous Flight

Joel Klimont*, Jakob Buchsteiner, Alexander Lampalzer and Radu Grosu
Technical University of Vienna, Austria

http://www.imavs.org/

ABSTRACT

Agile autonomous quadrotors require high-rate, accurately timestamped data exchange between the embedded flight controller and an onboard computer (OBC). Betaflight provides responsive rate control, DShot motor support, and mature gyroscope filtering, but its MultiWii Serial Protocol (MSP) interface is designed primarily for configuration and telemetry rather than for synchronized, high-rate transfer of low-level state to an onboard computer. We present *Weitflug*, a research-oriented fork of Betaflight that introduces the Flyby Serial Protocol (FSP) and RPM-aware accelerometer filtering. FSP periodically transmits fixed-format packets containing flight-controller timestamps, filtered inertial measurements, attitude, pilot inputs, motor speeds, and battery voltage. An OBC-side clock estimator maps flight-controller timestamps into the OBC time base. On a SPEEDYBEEF7V3 flight controller connected to a Jetson Orin NX, FSP provided a sensor stream at approximately 1.05 kHz. The 1st-to-99th-percentile spread of consecutive sample intervals was 0.007 ms, compared with 3.313 ms for MSP messages timestamped upon reception by the onboard computer. Spectral analysis of flight logs further showed strong attenuation of motor-synchronous accelerometer components after RPM-aware filtering. These results demonstrate a compact interface with flight-controller-side timestamps for using Betaflight as a low-level controller in autonomous multirotor research.

1 INTRODUCTION

Agile autonomous quadrotor flight places strict timing requirements on the sensing-to-actuation pipeline. At high speed, delayed or irregularly timed inertial measurements can degrade estimation and control, while limited access to low-level flight-controller data complicates system integration and analysis [1, 2, 3].

General-purpose autopilot stacks provide mature mission handling and companion-computer interaction, commonly through MAVLink [4, 5]. Betaflight occupies a different design point, emphasizing responsive FPV flight, high-rate rate control, DShot motor interfaces, and a mature tuning workflow [6]. Out of the box, however, it offers little support for synchronized high-rate state exchange with an onboard computer.

This paper presents *Weitflug*, a research-oriented fork of Betaflight[1] that retains Betaflight's embedded control stack while extending it for agile autonomous flight with an onboard computer (OBC). Our main contributions are:

- We introduce the *Flyby Serial Protocol* (FSP), a periodic serial interface with flight-controller-side sample timestamps and OBC-side clock alignment for high-rate state exchange.
- We implement RPM-aware accelerometer filtering by extending Betaflight's motor-speed-dependent notch-filtering infrastructure to acceleration measurements.
- We evaluate the timing information available to downstream software and the attenuation of motor-synchronous accelerometer components on a real flight-controller and onboard-computer setup.

To support reproducibility, the accompanying repository provides the firmware, OBC-side FSP implementation and clock estimator, build configuration, and analysis scripts used for the evaluation.

2 RELATED WORK

2.1 Autopilot and Research Architectures

Research UAV software spans different abstraction levels. PX4 provides a modular embedded robotics framework, Paparazzi an academic airborne and ground-control ecosystem, and ROSflight separates flight-controller firmware from ROS software on a Linux companion computer [4, 7, 8]. Complete platforms such as MRS and Agilicious additionally cover broader autonomy stacks, including estimation, control, simulation, perception, and hardware integration [9, 2]. Weitflug is complementary to these systems: it retains Betaflight's rate controller, receiver path, ESC support, tuning workflow, and filtering pipeline while providing a narrow embedded interface for independently developed OBC software.

*Email address(es): joel.klimont@tuwien.ac.at

[1] Open-source project repository: `https://github.com/F-WuTS/weitflug-open`

http://www.imavs.org/

Within Betaflight-derived research firmware, INDIflight improves UART telemetry and uplink for offboard estimation and control, adds alternative embedded control architectures, and includes an optional accelerometer RPM filter [10]. Betalink modifies MSP to exchange motor commands and flight-controller state in one transaction and adds an FC-side motor-velocity controller using bidirectional DShot feedback [11]. Weitflug instead focuses on a periodic, fixed-format state stream with source timestamps, OBC-side clock alignment, and an experimentally evaluated accelerometer-filtering path while retaining Betaflight's rate controller. Because these projects expose different control abstractions and communication semantics, we compare them qualitatively rather than treating their reported rates as directly interchangeable performance baselines.

2.2 Serial Interfaces and Measurement Timing

MAVLink is the dominant messaging protocol in autopilot-centered UAV systems and supports telemetry, commands, mission services, and offboard control [5]. Betaflight commonly uses the MultiWii Serial Protocol (MSP) for configuration and selected telemetry functions [12]. MSP is practical for these purposes, but obtaining the low-level measurements needed by an OBC controller requires multiple command-specific requests and responses with different payloads and update rates. A MAVLink-based interface would provide broader interoperability, whereas this work requires a compact, fixed-layout state-and-command exchange with explicit source-timestamp semantics. These requirements motivate the specialized recurring packet format described in Section 4.

Timing is part of the sensing interface, not merely an implementation property. Cheng et al. show that bounded sensing-to-actuation timing and measurement freshness are important in a real-time multirotor controller [1]. Temporal-calibration methods for visual–inertial estimation explicitly estimate sensor time offset because timestamp error degrades fusion consistency, and processing-dependent offsets may vary over time [13, 14]. To handle this, distributed synchronization work commonly models clock offset and skew and estimates them using a recursive filter such as a Kalman filter [15]. Weitflug applies this model to flight-controller timestamps and OBC reception times. This allows the onboard computer to reconstruct when each sample was recorded by the flight controller.

2.3 Motor-Synchronous Vibration and Filtering

Motor and propeller vibration is directly observable in onboard inertial measurements. Ferede et al. report that propeller-induced vibration can severely degrade accelerometer measurements on racing quadrotors and therefore use mechanical vibration damping in their experiments [16]. Flight stacks track rotor-related frequencies rather than relying only on broad low-pass filtering. Betaflight uses bidirectional DShot RPM telemetry to place gyroscope notches at motor fundamentals and harmonics, while ArduPilot supports throttle-, RPM-sensor-, ESC-telemetry-, and in-flight-FFT-based dynamic harmonic-notch tracking for gyroscope signals [17, 18]. Weitflug uses per-motor RPM measurements already available through Betaflight's bidirectional DShot path, applies the resulting notch structure to the accelerometer signal, and exports both filtered acceleration and motor-speed measurements to the OBC. The present evaluation demonstrates spectral attenuation of motor-synchronous components, while estimator benefit, filter delay, and computational cost remain to be quantified.

3 CURRENT BETAFLIGHT ARCHITECTURE AND LIMITATIONS

Betaflight is designed primarily for manual first-person-view (FPV) flight, where the flight controller executes a low-latency inner control loop and the pilot provides high-level commands. In this setting, the gyroscope is the dominant feedback signal for rate control, while communication with external systems is mainly used for configuration, telemetry, and debugging. When Betaflight is used as a low-level platform for autonomous flight, two architectural limitations become apparent: the timing behavior of its serial interface and the limited filtering support for accelerometer measurements.

3.1 MSP-Based Communication

Betaflight commonly communicates with external software through the MultiWii Serial Protocol (MSP). MSP is a request-response protocol: to acquire data such as IMU measurements, the onboard computer (OBC) sends a request to the flight controller (FC). The FC processes incoming requests in the serial task, which polls the UART interface at a rate configured by `serial_update_rate_hz`. Once a request is handled, the FC sends the corresponding response back to the OBC.

This design is convenient for configuration and telemetry, but it is not ideal for deterministic high-rate sensor streaming. Since data is only produced in response to OBC requests, jitter in the OBC request loop, kernel scheduling, UART transmission, or FC-side serial polling directly affect the timing of the received data. If the OBC requests data faster than the serial task processes incoming packets, repeated samples or irregular update intervals may occur.

A further limitation is timestamping. MSP packets are typically timestamped on the OBC when they are received. The timestamp therefore represents packet arrival time, not the sensor acquisition time on the FC. Delays from serial-task scheduling, UART transmission, kernel buffering, interrupt batching, and userspace scheduling are included in the timestamp. As a result, packets generated at regular intervals on the FC may appear irregular on the OBC. Conversely, packets delivered together by the kernel may receive nearly identical timestamps although they were transmitted at different times. Different MSP packet sizes also lead to different

http://www.imavs.org/

UART transmission times, which further distorts the observed inter-arrival intervals.

3.2 Accelerometer Filtering

Betaflight's sensor filtering pipeline is optimized mainly for gyroscope data. This reflects the requirements of manual FPV flight, where the gyro is the primary signal for the inner rate controller. Betaflight therefore provides extensive gyro filtering, including motor-RPM-aware filtering to suppress vibration components caused by motors and propellers [17].

The accelerometer is less central to typical FPV acro-mode flight and is therefore not filtered with the same emphasis. However, for autonomous flight, the accelerometer can be important for state estimation, inertial sensing, and higher-level control. Motor-induced vibrations are clearly visible in accelerometer measurements and can degrade estimator quality. Since these vibration frequencies depend on motor RPM, the lack of RPM-aware accelerometer filtering limits the usefulness of the accelerometer signal for onboard autonomy.

These limitations motivate Weitflug's two main extensions: the Flyby Serial Protocol (FSP), which periodically sends FC-timestamped state to the OBC, and RPM-aware filtering for accelerometer measurements.

4 WEITFLUG SYSTEM DESIGN

Weitflug adds a narrow research interface to flight controllers running Betaflight without changing the division of responsibilities between the embedded controller and the OBC. The flight controller continues to run IMU acquisition, filtering, attitude estimation, rate control, receiver handling, ESC communication, and motor output. The OBC runs experiment-specific control, estimation, and planning software and receives low-level measurements with flight-controller timestamps.

The exported state is restricted to quantities useful for agile-flight research: filtered accelerometer and gyroscope data, attitude, RC inputs, motor RPM, battery voltage, and timing metadata. This keeps the microcontroller-side interface compact while exposing the signals needed for estimator validation, controller debugging, and post-flight analysis.

4.1 Flyby Serial Protocol

For this purpose, we developed the *Flyby Serial Protocol* (FSP) as Weitflug's runtime serial interface. FSP transfers high-rate state and commands between the flight controller and the OBC, while MSP remains available for configuration. In contrast to MSP, where different pieces of state are distributed across several message types, FSP transmits one recurring packet containing the low-level state needed by the OBC.

FSP uses a fixed binary layout within each compiled protocol variant. As shown in Figure 1, the base FC-to-OBC and OBC-to-FC packets occupy 116 bytes and 24 bytes, respectively. Packets are COBS-encoded and terminated by a zero delimiter [19]. A CRC-8 with initial value `0xFF` and polynomial `0xD5` covers the unencoded packet except for its final checksum byte.

Each FC-to-OBC packet batches two sensor subframes to reduce packet-handling overhead. Each sensor subframe carries its own sample timestamp, filtered accelerometer measurement, accelerometer $1g$ scale, filtered gyroscope measurement and scale, quaternion attitude estimate, current RC values, four motor RPM values, and battery voltage. Including the accelerometer scale in every frame allows the current protocol to use a single self-contained message type, at the cost of retransmitting a configuration value. In future versions, we will explore separating configuration and measurement packets to reduce the transmission of redundant information. The OBC-to-FC payload contains commanded roll, pitch, yaw, and throttle values together with two auxiliary channels. After length, version, COBS, and CRC checks, these values enter Betaflight through the existing MSP receiver path and are interpreted as RC channels.

FSP adopts a periodic, flight-controller-initiated approach, in contrast to the request–response model of MSP. Sensor values are collected and timestamped from the PID-loop path and placed in a five-entry circular queue; two frames are removed for each transmitted packet. The default lockstep configuration sets an upper target of 1100 Hz and selects an integer PID-loop interval that does not exceed it. If the queue is full, the current implementation drops the new sensor frame without transmitting an explicit overflow counter.

For systems that also need to relay MAVLink traffic, FSP provides an optional compile-time tunnel. In this variant, a fixed 64-byte field in each FC-to-OBC packet is reserved for relaying MAVLink messages, increasing the packet size from 116 bytes to 180 bytes. A flag in the version/reserved field identifies the variant. The tunnel is auxiliary to the high-rate state exchange and is not evaluated further in this work.

4.2 Clock Alignment

The flight controller and OBC maintain independent clocks. On the OBC side, their relation is modeled as a linear map

$$t_{o,k} = \alpha_k t_{f,k} + \beta_k, \tag{1}$$

where $t_{f,k}$ is the flight-controller time, $t_{o,k}$ the corresponding OBC time, α_k the relative clock-rate ratio, and β_k the offset. After the first packet is received, both clocks are expressed relative to that reference sample. The Kalman filter state is

$$x_k = \begin{bmatrix} \alpha_k & \beta_k \end{bmatrix}^\top, \tag{2}$$

and is propagated with a random-walk process model

$$x_{k|k-1} = x_{k-1|k-1} + w_k, \qquad w_k \sim \mathcal{N}(0, Q). \tag{3}$$

For a received timestamp pair, the observation model is

$$z_k = H_k x_k + v_k, \qquad H_k = \begin{bmatrix} t_{f,k} & 1 \end{bmatrix}, \tag{4}$$

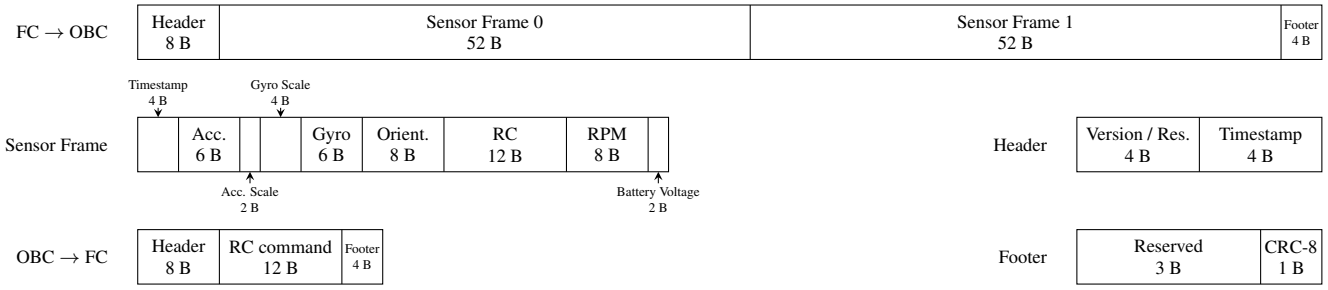


Figure 1: Logical layout of the base FSP packets in the current implementation. The FC-to-OBC packet batches two sensor subframes in one transmission, while the OBC-to-FC packet carries the current RC command set.

with v_k capturing transport delay variation and timestamping noise. In the current implementation, the filter uses $Q = \mathrm{diag}(10^{-10}, 10^{-6})$ and $R = 10^{-3}$, and every 100th received packet is used as an update sample. If the flight-controller timestamp decreases, the estimator resets; if the residual exceeds 0.5 s, which occurs for example when the flight-controller clock is paused during calibration, the reference times are reinitialized.

Once the filter has converged, the corrected OBC time of a packet is predicted from the packet-level timestamp. The two batched sensor samples are then assigned corrected timestamps by subtracting their within-packet offset in the flight-controller time base. This keeps clock estimation on the OBC and avoids adding a two-way synchronization exchange to the FC, while providing downstream estimation and control software with a common time base.

4.3 Integrity and Failure Handling

Packet validity is treated as part of the measurement interface. The receiver rejects packets with unexpected length, an unsupported protocol variant, failed COBS decoding, or an invalid CRC. FSP does not currently transmit a packet sequence number or queue-overflow counter. The OBC can nevertheless infer missing sensor frames from discontinuities in the embedded sample timestamps, but it cannot directly distinguish a queue overflow from other causes of loss.

4.4 RPM-Aware Accelerometer Filtering

Motor-synchronous vibration appears in both gyroscope and accelerometer measurements. Betaflight already uses RPM telemetry to place gyro notches at motor fundamentals and harmonics. Weitflug instantiates the same notch-filter implementation a second time for the three accelerometer axes and returns the filtered values together with motor RPM through FSP. The accelerometer configuration supports up to three harmonics; its current defaults enable the fundamental for each motor, use a minimum frequency of 60 Hz, a 50 Hz fade range, and a notch quality factor of 5. This targets motor-synchronous acceleration components without requiring a broadly lower low-pass cutoff.

5 EVALUATION

The evaluation benchmarks the two main extensions in Weitflug. First, we compare the timing behavior available to downstream software when transferring IMU data from the flight controller to the OBC using FSP and MSP. We then qualitatively assess the effect of RPM-aware accelerometer filtering using real flight logs.

5.1 Experimental Setup

The test platform consists of a SPEEDYBEEF7V3 flight controller with an STM32 F7 processor and a Jetson Orin NX 16GB as the OBC, connected through MSP or FSP. The flight controller is configured for a nominal control-loop frequency of 3.2 kHz; the scheduler reports an actual frequency of 3148 Hz. We additionally configured the Betaflight parameter `serial_update_rate_hz` to 1000 Hz, because MSP messages were otherwise not processed responsively enough for the timing analysis. Acceleration is measured by the BMI270 six-axis IMU on the flight controller, and the electronic speed controller provides the per-motor RPM measurements used for filtering.

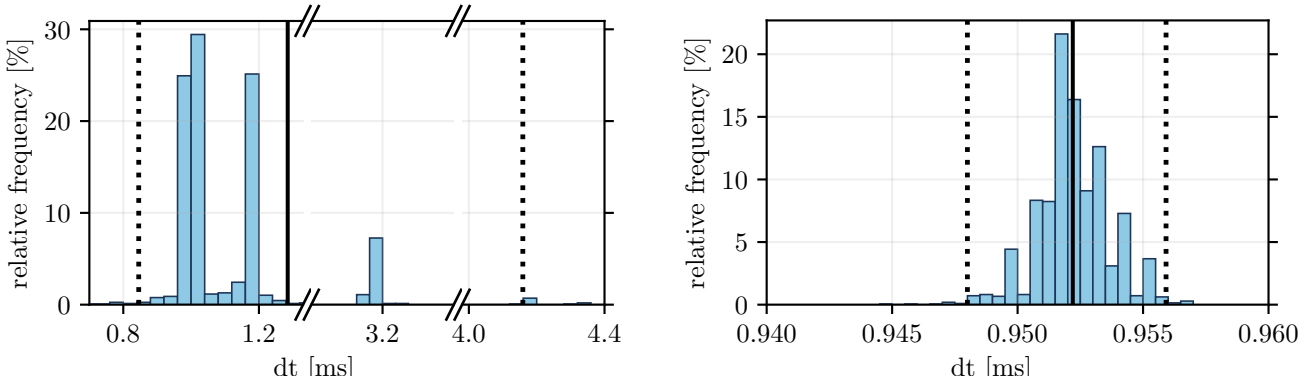
5.2 Timing Analysis of FSP and MSP

For FSP, the OBC records both packet-reception times and decoded, clock-aligned sensor-frame timestamps. For MSP, we request IMU data at 1000 Hz and use reception times because the protocol responses do not include the corresponding flight-controller sample timestamps. We additionally request data also carried by FSP, including RC commands and attitude at 100 Hz and battery voltage at 10 Hz, to make the serial workload more representative. The evaluated firmware sets the lockstep target to 1100 Hz. With the measured 3148 Hz PID-loop frequency, the integer lockstep interval is three iterations, yielding a nominal source-sample rate of approximately $3148/3 = 1049.3$ Hz.

The MSP result should therefore be interpreted as an application-level baseline under a representative mixed telemetry workload, rather than as the maximum standalone IMU throughput achievable with MSP.

We report the 1st-to-99th-percentile spread of consecutive timestamp deltas and refer to this quantity as jitter. Table 1

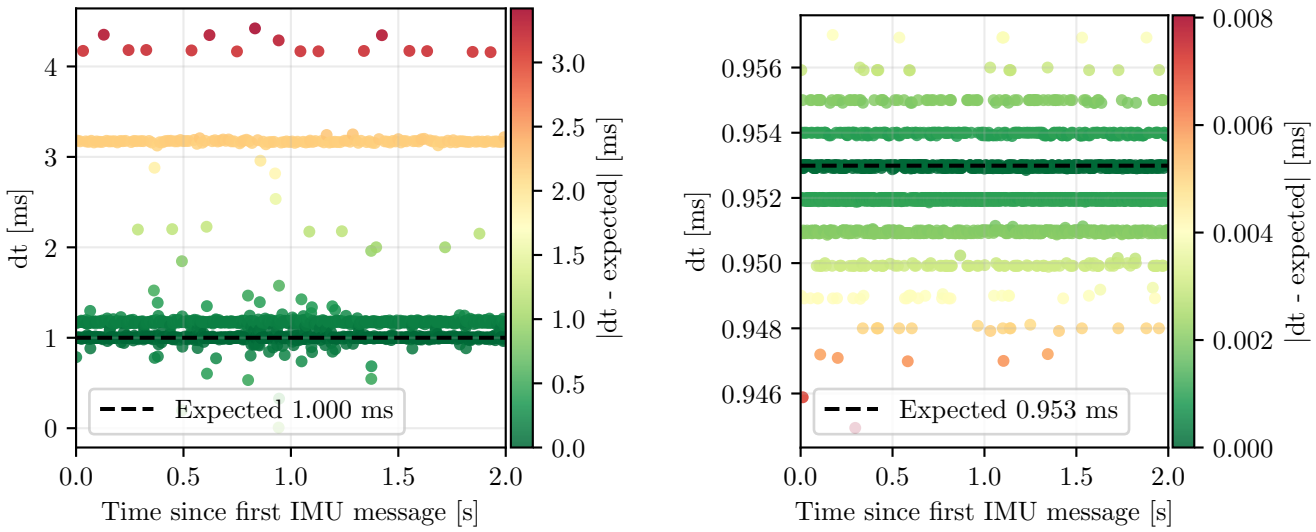
http://www.imavs.org/



(a) Betaflight with MSP

(b) Weitflug with FSP

Figure 2: Distribution of timestamp delta times between MSP and FSP. The solid line indicates the mean, while the dotted lines denote the 1st and 99th percentile. The MSP distribution has multiple modes, one around the expected interval of 1 ms, and three additional ones around 1.2 ms, 3.2 ms and 4 ms. FSP shows a narrow, quantized distribution around the expected interval, with the remaining spread attributed to flight-controller scheduling and timer quantization.

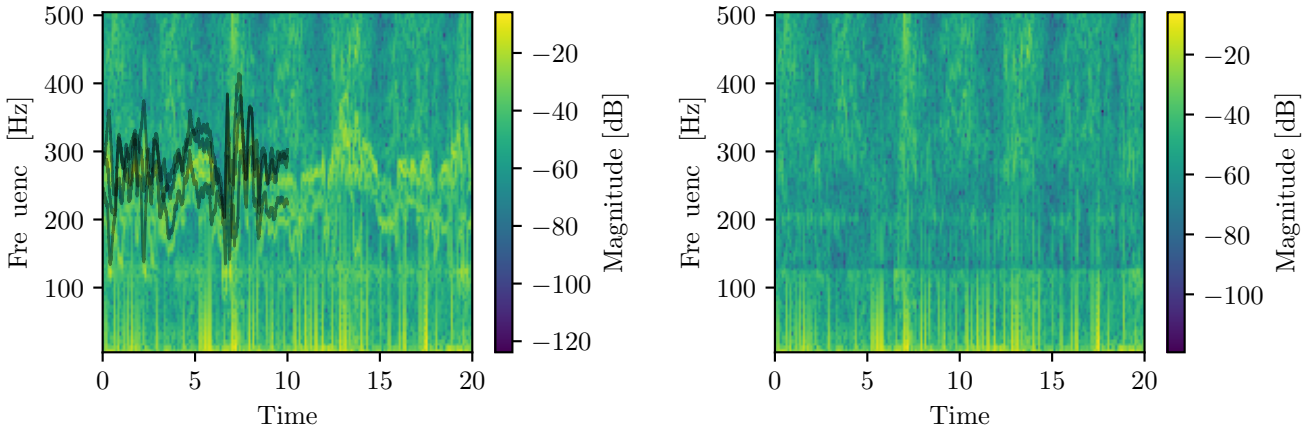


(a) Betaflight with MSP

(b) Weitflug with FSP

Figure 3: Timestamp deltas between consecutive IMU messages for MSP and FSP. Subfigure (a) shows periodic MSP reception-time outliers of up to approximately four times the expected interval, whereas they are absent from the FSP source-time sequence in subfigure (b). MSP timestamps are captured on reception by the OBC, while FSP timestamps originate on the flight controller; the plots therefore show the timing information available to downstream software rather than an equal-condition measurement of transport latency.

http://www.imavs.org/



(a) Unfiltered accelerometer with overlaid motor speed in revolutions per second (black, first half only)

(b) RPM-aware filtered accelerometer

Figure 4: Short-time Fourier analysis of accelerometer vibration before and after RPM-aware filtering. The unfiltered spectrum exposes motor-synchronous bands whose time-varying frequency follows the RPM trajectory. The filtered spectrum suppresses these bands, reducing motor-induced components in the accelerometer signal. Sampling frequency 1000 Hz, 0.2 s Hann window with 50% overlap.

Table 1: Timestamp deltas between consecutive IMU samples

Metric	MSP [ms]	FSP [ms]
Expected	1.000	0.953
Mean	1.284	0.952
Median	1.003	0.952
1st percentile	0.845	0.948
99th percentile	4.159	0.956
P99–P1 spread	3.313	0.007

and figs. 2 and 3 summarize the results. Under the timestamping conventions above, the FSP source-time sequence has substantially lower interval spread than the MSP reception-time sequence. This comparison quantifies the regularity of the sample timeline available to downstream software; it does not measure or remove the constant component of one-way transport latency.

FSP reports a mean rate of $f_{\mathrm{mean,FSP}} = 1/dt_{\mathrm{mean,FSP}} = 1050$ Hz. The small difference from the selected 1049 Hz rate may arise from clock-rate mismatch, timer quantization, and rounding in the synchronization estimate; a dedicated clock-accuracy evaluation is left for future work. MSP reaches a mean reception rate of $f_{\mathrm{mean,MSP}} = 1/dt_{\mathrm{mean,MSP}} = 778$ Hz in this configuration.

5.3 *RPM-Aware Accelerometer Filtering*

The accelerometer filter is evaluated from flight logs containing both unfiltered and filtered acceleration signals. Figure 4 compares short-time Fourier transforms of the acceleration magnitude over the same motor-speed sweep. In the unfiltered signal, motor-synchronous ridges remain sufficiently pronounced that the RPM trajectory can be read directly from the spectrum. After RPM-aware filtering, these ridges are strongly attenuated and no longer provide a reliable RPM trace. This result demonstrates attenuation of the dominant motor-synchronous components; quantifying total noise reduction, estimator impact, and filter delay remains future work.

6 Conclusion

We presented Weitflug, a research-oriented Betaflight fork that provides periodic, flight-controller-timestamped state through FSP and extends RPM-aware filtering to accelerometer measurements. In our evaluation, the 1st-to-99th-percentile spread of consecutive IMU intervals decreased from 3.313 ms for MSP reception timestamps to 0.007 ms for the FSP source-time sequence while maintaining a stream above 1000 Hz. Flight-log analysis also showed attenuation of motor-synchronous accelerometer components. Together, these extensions make Betaflight more suitable as a low-level controller for autonomous multirotor research.

The current evaluation does not measure absolute one-way transmission latency, and the one-way clock-alignment method cannot separate a constant transport delay from clock offset. This distinction is especially relevant when FSP timestamps are fused with measurements from sensors connected directly to the OBC. Future work should evaluate latency us-

http://www.imavs.org/

ing an external reference or two-way exchange and quantify accelerometer-filter attenuation, phase delay, computational cost, and estimator impact over multiple flights. Timestamped command queuing could additionally improve input–output timing repeatability.

REFERENCES

[1] Zhuoqun Cheng, Richard West, and Craig Einstein. End-to-end analysis and design of a drone flight controller. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2404–2415, 2018.

[2] 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. *Science Robotics*, 7(67):eabl6259, 2022.

[3] 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.

[4] Lorenz Meier, Dominik Honegger, and Marc 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.

[5] Anis Koubaa, Azza Allouch, Maram Alajlan, Yasir Javed, Abdelfettah Belghith, and Mohamed Khalgui. Micro Air Vehicle Link (MAVLink) in a Nutshell: A Survey. *IEEE Access*, 7:87658–87680, 2019.

[6] Betaflight Development Team. Betaflight Documentation. Online documentation. Accessed: 2026-06-15.

[7] Balazs Gati. Open source autopilot for academic research: The Paparazzi system. In *2013 American Control Conference*, pages 1478–1481, 2013.

[8] James Jackson, Gary J. Ellingson, and Timothy W. McLain. ROSflight: A lightweight, inexpensive MAV research and development tool. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 758–762, 2016.

[9] Tomáš Báča, Matěj Petrlík, Matouš Vrba, Vojtěch Spurný, Robert Pěnička, 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, 2021. Article 26.

[10] INDIflight Development Team. INDIflight: Research-focused fork of betaflight 4.4.2. Open-source repository. Accessed: 2026-07-24.

[11] Betalink Development Team. Betalink: Low-latency communication with betaflight. Open-source repository. Accessed: 2026-07-24.

[12] Betaflight Development Team. MSP Protocol Reference for Developers. Online documentation. Accessed: 2026-06-15.

[13] Mingyang Li and Anastasios I. Mourikis. Online temporal calibration for camera-IMU systems: Theory and algorithms. *The International Journal of Robotics Research*, 33(7):947–964, 2014.

[14] Yonggen Ling, Linchao Bao, Zequn Jie, Fengming Zhu, Ziyang Li, Shanmin Tang, Yongsheng Liu, Wei Liu, and Tong Zhang. Modeling varying camera-IMU time offset in optimization-based visual-inertial odometry. In *Computer Vision - ECCV 2018*, pages 491–507. Springer International Publishing, 2018.

[15] Brent R. Hamilton, Xiaoli Ma, Qiang Zhao, and Jun Xu. ACES: Adaptive clock estimation and synchronization using Kalman filtering. In *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*, MobiCom '08, pages 152–162, 2008.

[16] Robin Ferede, Till Blaha, Erin Lucassen, Christophe De Wagter, and Guido C.H.E. De Croon. One net to rule them all: Domain randomization in quadcopter racing across different platforms. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6357–6363, 2025.

[17] Betaflight Development Team. DShot RPM Filtering. Online documentation. Accessed: 2026-06-15.

[18] ArduPilot Development Team. Managing gyro noise with dynamic harmonic notch filters. Online documentation. Accessed: 2026-06-15.

[19] Stuart Cheshire and Mary Baker. Consistent overhead byte stuffing. *IEEE/ACM Transactions on Networking*, 7(2):159–172, 1999.

http://www.imavs.org/

Flight-Ready LiDAR-Inertial Odometry for Embedded Drone Platforms

Alvaro J. Gaona, David Perez-Saura, Francisco J. Anguita, and Pascual Campoy
Computer Vision and Aerial Robotics Group at Centre for Automation and Robotics C.A.R. (UPM-CSIC), Universidad Politécnica de Madrid (CVAR-UPM), Calle Jose Gutierrez Abascal 2, 28006 Madrid, Spain

http://www.imavs.org/

Abstract

Open-source LiDAR–inertial odometry (LIO) systems have achieved remarkable benchmark accuracy, yet current state-of-the-art implementations are primarily optimized for evaluation performance rather than the requirements of real-time closed-loop aerial control. When deployed onboard UAVs, this can introduce limitations that degrade flight performance. In this work, we identify five architectural deficiencies in a representative tightly coupled IESKF-based LIO implementation: odometry publishing tied to the LiDAR rate (10 Hz instead of the IMU's 200 Hz), missing velocity outputs, execution bottlenecks that block IMU processing, mutex contention, and synchronization race conditions. We introduce corresponding modifications including IMU-rate forward propagation, direct body-frame velocity publishing, SLERP-based smoothing, dual-executor isolation, and explicit synchronization protection. The resulting system increases odometry output from 10 Hz to a stable 200 Hz, provides a complete Twist state at every IMU sample, and preserves continuity during transient LiDAR loss. Experiments on a Livox Mid-360 / Pixhawk 4 Mini autonomous UAV with motion-capture ground truth validate the approach. Since the underlying estimator (IESKF + ikd-Tree) remains unchanged, the proposed improvements can be directly applied to FAST-LIO2-derived implementations.

1 Introduction

Tightly-coupled LiDAR-inertial odometry has matured rapidly over the last five years, with implementations such as FAST-LIO2 [1], LIO-SAM [2], Point-LIO [3], and DLIO [4] routinely reporting centimeter-level accuracy on standard benchmark datasets. These implementations are open-source, well-documented, and widely cited; a roboticist building an aerial-mapping system has every reason to start from one of them.

A roboticist building an autonomous *flight* system, however, quickly discovers that benchmark accuracy and flight-readiness are not the same thing. A flight controller imposes requirements that benchmark evaluations do not exercise: pose feedback at hundreds of Hertz to keep the inner control loop stable, linear and angular velocity estimates at the same rate for damping and feedforward, low-noise signals that do not excite actuator oscillations, and uninterrupted continuity even during transient sensor degradation. An LIO implementation that meets none of these requirements out of the box is not unusable; it is simply not yet ready for flight.



Figure 1: Quadrotor platform used for the validation flights: Livox Mid-360 LiDAR on top, Pixhawk 4 Mini autopilot and Jetson Orin NX onboard computer below, reflective mocap markers on the prop guards. Flight footage: `https://vimeo.com/1192528650?share=copy&fl=sv&fe=ci`.

This paper documents the modifications required to take a representative open-source IESKF-based LIO implementation from benchmark-tuned to flight-ready.[1] The five contributions are:

1. A forward-propagation scheme that publishes pose and velocity at every IMU sample (200 Hz) using the most recent IESKF-corrected state as an anchor.

2. Body-frame linear and angular velocity derived directly from the filter state, populating the previously-empty field of the odometry message.

3. A first-order EMA filter on velocity and SLERP-based filtering on orientation, with cutoff frequencies chosen to remove integration noise without obscuring drone-bandwidth maneuvers.

[1] Source code: `https://github.com/alvgaona/fr-lio`.

4. A dual-executor architecture that isolates the IMU callback from scan-processing latency, eliminating rate-degradation under load.

5. An identification and fix of a race condition in the IMU/LiDAR synchronization function that emerges only under multi-threaded execution.

These modifications are purely architectural: the underlying IESKF state propagation, point-to-plane registration, and ikd-Tree map remain unchanged. They apply to any FAST-LIO2-derived implementation and to any IESKF-based system where the filter correction operates at a rate slower than the IMU.

The paper is organized as follows. Section 2 surveys representative open-source LIO implementations and characterizes the flight-readiness gap each leaves open. Section 3 reviews the publishing pipeline of the baseline implementation and identifies the limitations that motivate each modification. Section 4 presents the IMU-rate forward propagation and body-frame velocity derivation. Section 5 describes the dual-executor architecture, mutex-contention reduction, and synchronization race-condition fix. Section 6 covers EMA/SLERP filtering and chained-propagation fallback for LiDAR-interruption robustness. Section 7 reports flight-test validation on a Livox Mid-360 / Pixhawk 4 Mini platform with motion-capture ground truth. Section 8 concludes.

2 RELATED WORK

Modern LIO implementations have converged on two architectural patterns: tightly-coupled iterated error-state extended Kalman filters (IESKF) with incremental map structures, and tightly-coupled factor graphs with incremental smoothing back-ends. Both deliver centimeter-accurate trajectories on benchmark datasets [1, 2, 3, 4]; neither was designed with the requirements of an in-flight controller in mind. We briefly review the four most widely-used representatives and characterize the flight-readiness gap each leaves open.

FAST-LIO2 [1] is the canonical IESKF+*ikd*-Tree implementation. The state is propagated forward by IMU integration over each scan interval, then corrected by a single IESKF iteration loop with point-to-plane registration against the *ikd*-Tree map. Publishing happens once per scan, at the end of the IESKF correction. The default ROS 1/2 deployment uses a single-threaded executor, so all callbacks (IMU buffering, LiDAR buffering, IESKF processing, publishing) serialize on one thread. The Twist field of the published `Odometry` message is left at zero. None of these choices is wrong for offline mapping or benchmark evaluation; all four limit flight-readiness directly.

LIO-SAM [2] replaces the IESKF with a tightly-coupled factor graph optimized incrementally via iSAM2. It supports GPS and loop-closure factors, which makes it well-suited to outdoor mapping. Publishing rate is again tied to the keyframe selection (typically 1–10 Hz), and the published velocity is derived from preintegrated IMU factors only at keyframe boundaries. The factor-graph back-end is computationally heavier than an IESKF and the per-keyframe variability makes the publishing rate unpredictable, both of which compound the flight-control problem rather than relieve it.

Point-LIO [3] addresses high-bandwidth motion by treating each LiDAR point as an individual measurement update rather than batching points into accumulated scans. This raises the IESKF correction rate substantially (the authors report up to kilohertz output) and is well-suited to aggressive motions where intra-scan distortion would otherwise dominate. The output frequency is high, but the architecture inherits its FAST-LIO2's single-executor concurrency model; under sustained heavy load the same thread still serializes IMU, LiDAR, and publishing. Point-LIO closes one of the four flight-readiness gaps (rate) without addressing the others (velocity availability, executor isolation, race-condition robustness).

DLIO [4] is a recent lightweight LIO that adds continuous-time motion correction to deskew scans during integration, packaged with a clean ROS 2 implementation. The lightweight design makes it attractive for embedded deployment, and the ROS 2 native architecture supports multi-threaded executors in principle. However, the published implementation does not by default isolate the IMU callback to a dedicated executor, does not populate the Twist field with body-frame velocities, and inherits the same publishing-coupled-to-LiDAR pattern as FAST-LIO2. The architectural improvements are at the estimation layer, not at the deployment layer.

Summary. Across all four representative implementations, the flight-readiness gap manifests as the same four issues: publishing rate coupled to (or capped by) the LiDAR rate, missing velocity fields, single-threaded execution that blocks the IMU callback under load, and an absence of any mechanism to maintain odometry continuity during transient LiDAR loss. Table 1 summarizes which axes each implementation addresses out of the box. The modifications presented in this paper close all four gaps simultaneously and apply directly to any of these implementations, since they intervene at the publishing layer rather than at the estimation layer; the underlying choice of IESKF vs. factor graph, batched scans vs. per-point updates, or discrete vs. continuous-time motion correction is independent of the architectural fix. Visual-inertial state estimators such as VINS-Mono [5] and OpenVINS [6] already publish at IMU rate by propagating the corrected state between visual updates; our contribution can be read as adapting that publishing pattern to LIO.

3 BACKGROUND AND LIMITATIONS

The baseline implementation publishes odometry inside the LiDAR scan-processing loop. Each cycle accumulates a complete scan, executes IMU forward propagation across

http://www.imavs.org/

	200 Hz rate	Velocity output	Isolated IMU thread	LiDAR-loss resilience
FAST-LIO2 [1]	no	no	no	no
LIO-SAM [2]	no	partial	no	no
Point-LIO [3]	yes	no	no	no
DLIO [4]	no	no	no	no
This work	**yes**	**yes**	**yes**	**yes**

Table 1: Flight-readiness axes addressed by representative open-source LIO implementations. "Velocity output" = body-frame linear and angular velocities published alongside the pose. "Isolated IMU thread" = IMU sampling runs on a dedicated thread, decoupled from scan-processing latency. "LiDAR-loss resilience" = mechanism to maintain odometry continuity when LiDAR scans stop arriving for a transient period.

the scan interval, performs IESKF iterations with point-to-plane registration, updates the ikd-Tree map, and publishes the resulting pose. The publishing rate is therefore tied to the LiDAR scan-arrival rate.

For the Livox Mid-360 sensor at 100 Hz, an accumulator collecting 10 packets per IESKF cycle yields a ∼10 Hz publishing rate. At a flight speed of 2 m/s, the controller receives a new pose every 0.2 m of travel and must extrapolate between updates—an extrapolation that is unsafe for tight closed-loop control. Three structural issues underlie this:

Publishing rate coupled to LiDAR rate. When scan processing exceeds the inter-scan period, for example during heavy ikd-Tree insertions, the publishing rate degrades below 10 Hz. The rate is therefore not a guarantee but a best-case.

Absence of velocities. The published odometry message carries only the pose; the velocity component is left at zero. The flight controller must numerically differentiate position to obtain velocity, amplifying high-frequency noise and introducing at least one sampling period of lag.

Single-threaded execution. The baseline serializes IMU buffering, LiDAR buffering, IESKF processing, and publishing on a single thread. While the IESKF processes a scan, the IMU samples are queued. When publishing happens at the end of scan processing this serialization is harmless; moving the publication to the IMU sample rate to achieve 200 Hz makes blocking the dominant bottleneck.

4 IMU-RATE FORWARD PROPAGATION

4.1 Operating Principle

The IESKF correction occurs at the accumulated LiDAR rate (∼10 Hz). Between corrections, each new IMU sample propagates the state forward using the inertial dynamics. The propagation does not replace the filter; it uses the most recent corrected state as an *anchor* and applies single-step integration with the current IMU measurement.

Let t_a denote the time of the last IESKF correction and $(\mathbf{R}_a, \mathbf{p}_a, \mathbf{v}_a, \mathbf{b}_{g,a}, \mathbf{b}_{a,a}, \mathbf{g}_a)$ the corrected anchor state. For an IMU sample at time $t > t_a$ with raw measurements $\boldsymbol{\omega}_m$ and $\mathbf{a}_m$, the bias-corrected angular velocity and body-frame acceleration are

$$\boldsymbol{\omega} = \boldsymbol{\omega}_m - \mathbf{b}_{g,a}, \quad \mathbf{a}_b = \mathbf{a}_m - \mathbf{b}_{a,a}, \tag{1}$$

and the world-frame acceleration is $\mathbf{a}_w = \mathbf{R}_a \mathbf{a}_b + \mathbf{g}_a$. With $\Delta t = t - t_a$, the propagated state is

$$\mathbf{p}(t) = \mathbf{p}_a + \mathbf{v}_a \Delta t + \tfrac{1}{2}\mathbf{a}_w \Delta t^2, \tag{2}$$

$$\mathbf{v}(t) = \mathbf{v}_a + \mathbf{a}_w \Delta t, \tag{3}$$

$$\mathbf{R}(t) = \mathbf{R}_a \,\mathrm{Exp}(\boldsymbol{\omega} \Delta t), \tag{4}$$

where $\mathrm{Exp} : \mathbb{R}^3 \to \mathrm{SO}(3)$ is the SO(3) exponential map. Equation (4) is the zeroth-order discretization, treating $\boldsymbol{\omega}$ as constant over Δt; this is appropriate at the 5 ms IMU period. The integration also assumes that $\mathbf{R}_a$, the biases, and gravity remain constant over Δt. For $\Delta t < 0.1$ s (the nominal 10 Hz IESKF rate), the resulting error is on the order of centimeters in position and fractions of a degree in orientation [7]—negligible relative to the controller's resolution.

4.2 Anchor Update

Each IESKF correction copies its output state to the propagation anchor under mutex protection. The anchor stores position, velocity, rotation, gyroscope and accelerometer biases, gravity, and the timestamp of the corresponding scan end. The next IMU sample resets Δt to a small value, so propagation never accumulates error beyond the inter-correction interval.

4.3 Body-Frame Velocities

The IESKF state vector includes velocity in the world frame, $\mathbf{v} \in \mathbb{R}^3$. The flight controller requires velocity in the body frame, obtained via the inverse rotation:

$$\mathbf{v}_b(t) = \mathbf{R}(t)^\top \mathbf{v}(t). \tag{5}$$

The angular velocity in the body frame is the bias-corrected gyroscope measurement, available directly from (1). Both quantities populate the previously-empty Twist field of the published `Odometry` message.

5 THREADING AND CONCURRENCY

5.1 Dual-Executor Isolation

To publish at 200 Hz, the IMU callback must execute without delay at every sample. A single-threaded executor cannot satisfy this when scan processing takes tens of milliseconds (typical during heavy ikd-Tree insertions): under load, the IMU callback is starved for the duration of every IESKF cycle and the effective publishing rate degrades to the LiDAR rate.

The fix uses two independent single-threaded executors. The IMU callback is registered to an exclusive callback group assigned to a dedicated executor that runs on its own thread;

http://www.imavs.org/

the LiDAR callback, IESKF timer callback, and point-cloud publishers are assigned to the main executor. The IMU thread is therefore decoupled from scan-processing time and contends only for shared mutexes, not for execution slots.

5.2 Mutex Contention from In-Buffer Preprocessing

Even with separate executors, the IMU thread can be blocked if it requires a mutex held by the main thread. In the baseline, the LiDAR callback acquires the buffer mutex *before* performing scan preprocessing (point filtering, format conversion), an operation that takes several milliseconds.

The fix is to perform preprocessing outside the mutex-protected section. The mutex is acquired only for the buffer insertion of the preprocessed scan, an operation that takes microseconds. This reduces mutex hold time by approximately three orders of magnitude.

5.3 Race Condition in Synchronization

The packet synchronization function pairs LiDAR scans with IMU measurements to form IESKF inputs. In the baseline, this function accesses shared buffers without mutex protection. Under single-threaded execution this is safe; under the dual-executor architecture, concurrent writes from the IMU callback and reads from the synchronization function constitute a data race that occasionally produces malformed input pairs and silently degrades estimator accuracy.

The fix protects the synchronization function with the same mutex that guards the buffers. The mutex critical section is short (single-pass over both buffers), and the operation does not appear in any benchmark trace because it is masked by IESKF processing.

5.4 Path Message Serialization

A secondary bottleneck arises from publishing the `nav_msgs::Path` message at the IMU rate. `Path` accumulates the full trajectory; at 200 Hz, after 30 s of flight the message contains 6000 poses and serialization consumes significant CPU on the IMU thread. The fix throttles `Path` publication to 10 Hz; the message is visualization data, not control data, and the rate reduction has no functional effect.

6 FILTERING AND ROBUSTNESS

6.1 EMA on Velocity and SLERP on Orientation

Single-step integration of accelerometer measurements transfers high-frequency noise directly to the propagated velocity, producing typical fluctuations of $\sim$0.2 m/s (and several-m/s peak excursions on the vertical axis, see Figure 4) that a finely-tuned velocity controller would amplify into actuator oscillations. We apply a first-order exponential moving average (EMA) to the published body-frame velocities:

$$\bar{\mathbf{v}}_b[k] = \alpha \mathbf{v}_b[k] + (1-\alpha)\bar{\mathbf{v}}_b[k-1], \tag{6}$$

with the same form applied to the body-frame angular velocity. The cutoff frequency at sampling rate f_s is $f_c = (f_s/2\pi)\ln(1/(1-\alpha))$. For $f_s = 200$ Hz and $\alpha = 0.05$,

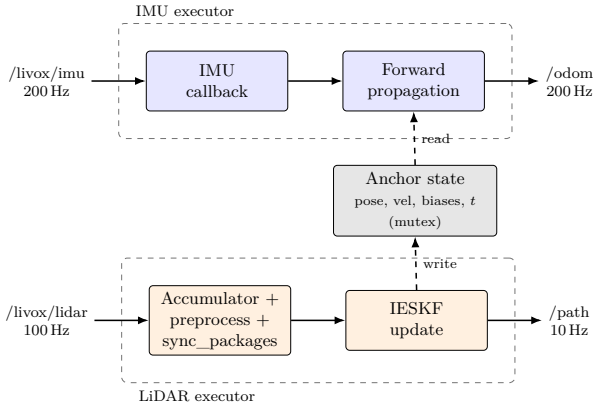


Figure 2: Dual-executor threading architecture. The IMU callback runs on its own single-threaded executor (top) and publishes the 200 Hz `/odom` stream by forward-propagating from the shared anchor state. LiDAR preprocessing and IESKF correction run on a second executor (bottom) and write the anchor on each successful update. The anchor state (pose, velocity, biases, timestamp) is mutex-protected; the critical section is short and the synchronization function acquires the same mutex.

$f_c \approx 1.6$ Hz, which removes accelerometer noise without significantly attenuating drone maneuvers under the precision-hover and indoor-trajectory regimes used in our validation (closed-loop bandwidth $\lesssim 1$ Hz). Aggressive flight regimes (racing-style trajectories, heavy payload swings) extend the controller bandwidth above this cutoff and require a larger α at the cost of higher noise pass-through.

Quaternion filtering cannot be performed component-wise. We use spherical linear interpolation (SLERP):

$$\bar{\mathbf{q}}[k] = \mathrm{SLERP}\left(\bar{\mathbf{q}}[k-1], \mathbf{q}[k], \alpha\right), \tag{7}$$

which respects the manifold structure of S^3 and reduces to weighted-and-normalized linear interpolation for small angular changes between consecutive samples.

Linear-velocity filtering is applied *after* the rotation to the body frame, not before; filtering in the world frame and then rotating injects gyroscope noise into the otherwise-clean filtered signal.

6.2 Chained Propagation under LiDAR Interruption

If LiDAR scans stop arriving (sensor obstruction, packet loss in the DDS network, transient driver failure), the anchor timestamp freezes and Δt grows without bound. For large Δt, the constant-rotation assumption of (2) breaks down and integration error becomes quadratic.

When Δt exceeds 0.5 s, the propagation is *chained*: the propagated state becomes the new anchor for subsequent IMU samples,

$$\mathbf{p}_a, \mathbf{v}_a, \mathbf{R}_a, t_a \leftarrow \mathbf{p}(t_k), \mathbf{v}(t_k), \mathbf{R}(t_k), t_k. \tag{8}$$

http://www.imavs.org/

Biases and gravity are not updated during chaining, since the IESKF estimate remains the best available. Each subsequent integration operates on a small Δt (the IMU period, 5 ms), avoiding quadratic error accumulation.

6.3 *Anchor Timestamp Guard*

When IESKF processing of a scan takes longer than the inter-scan period, the IESKF may attempt to overwrite the anchor with a state whose timestamp is earlier than what chained propagation has already advanced to. This would produce a backward jump in the published odometry. The fix conditions the write of position, velocity, rotation, and timestamp on the new timestamp being later than the current anchor. Biases and gravity are always updated, since they are independent of the time index.

7 FLIGHT-TEST VALIDATION

We validate the modifications on the quadrotor shown in Figure 1: a Livox Mid-360 LiDAR, a Pixhawk 4 Mini autopilot, and a Jetson Orin NX running ROS 2 (Humble), with a takeoff weight of approximately 1.8 kg and a 4S 5000 mAh battery. Validation flights were conducted in the Drone Arena of the Centre for Automation and Robotics (CAR), Universidad Politécnica de Madrid, operated by the Computer Vision & Aerial Robotics (CVAR) Group. The arena is a $6 \times 6 \times 12$ m volume instrumented with 16 OptiTrack motion-capture cameras providing 6-DOF ground-truth pose at 100 Hz with submillimeter accuracy.

The flight stack is built on the Aerostack2 framework [8], which provides the high-level mission management, behaviour layer, and state-machine orchestration. The inner-loop position and attitude controller is a differential-flatness controller [9] that exploits the quadrotor's flatness property to compute attitude and thrust setpoints from desired position trajectories and their derivatives; this controller is the principal consumer of the high-rate odometry and velocity stream produced by the modifications described in this paper. Two autonomous indoor flight sequences are used in what follows. A 58 s sequence, replayed under identical inputs through both the baseline FAST-LIO and the modified implementation, supports the publishing-rate histogram and the trajectory-accuracy comparison (Figure 3, Table 2). A separate 60 s sequence supports the body-frame velocity tracking analysis (Figure 4); the same sequence, with a 1.5 s contiguous block of `/livox/lidar` messages removed offline before replay through both implementations, is used to exercise the chained-propagation fallback (Figure 5).

7.1 *Publishing Rate*

Figure 3 shows the inter-message interval distribution for the published odometry topic, comparing the baseline against the modified implementation. The modified implementation publishes at a mean interval of 5.00 ms ($\sigma = 0.90$ ms, $p_{99} = 6.95$ ms) — a steady 200 Hz — while the baseline distribution

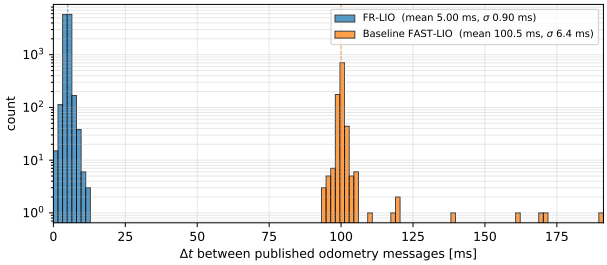


Figure 3: Inter-message interval distribution for the published odometry topic on a log-scale y-axis. The modified implementation (blue) concentrates at $\Delta t = 5$ ms ($f = 200$ Hz, $\sigma = 0.90$ ms), while baseline FAST-LIO (orange) publishes at the scan rate near $\Delta t = 100$ ms ($\sigma = 6.4$ ms).

sits at 100.5 ms ($\sigma = 6.4$ ms, $p_{99} = 107.8$ ms), reflecting its scan-rate publishing.

7.2 *Trajectory Accuracy*

Metric	Baseline	Modified
ATE RMSE (cm)	11.7	5.2
ATE median (cm)	4.7	3.8
ATE max (cm)	32.4	28.3
RPE 1 s (cm)	4.9	2.4

Table 2: Trajectory accuracy against motion-capture ground truth on a 58 s indoor flight. Each LIO trajectory is rigidly aligned to ground truth via least-squares SE(3) over the first 5 s of motion (motion onset detected at GT speed > 0.1 m/s sustained for 0.3 s).

Table 2 compares the published trajectory against motion-capture ground truth. The modified implementation matches the baseline on median ATE within 1 cm while approximately halving the RPE (1 s) drift, reflecting the tighter timing of the dual-executor architecture without altering the iterated Kalman update. The ATE max values (28–32 cm) for both implementations are isolated transients during rapid yaw segments where the IESKF registration is briefly mis-constrained by the Mid-360's reduced lateral overlap; the median and RMSE are the load-bearing comparison numbers.

7.3 *Velocity Tracking*

Figure 4 compares published body-frame velocities against ground-truth velocities derived from motion-capture position with a zero-phase Savitzky–Golay smoother (21-tap, order 3). Over the 60 s flight, the EMA-filtered output tracks ground truth with RMSE 0.05 m/s on the horizontal axes (v_x, v_y) and 0.10 m/s on v_z. The high-frequency content removed by the EMA, measured as the standard deviation of the unfiltered–filtered difference, is 5–11 cm/s per

http://www.imavs.org/

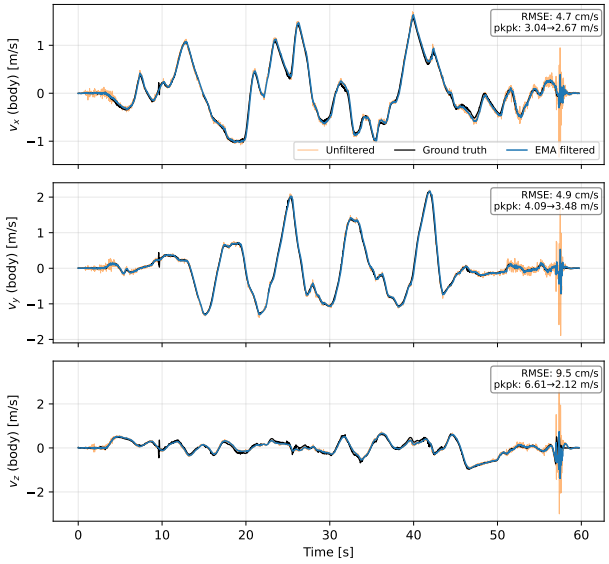


Figure 4: Body-frame linear-velocity tracking against motion-capture ground truth. EMA filter ($\alpha = 0.05$, $f_c \approx 1.6$ Hz) removes accelerometer noise without lagging the controller-bandwidth signal.

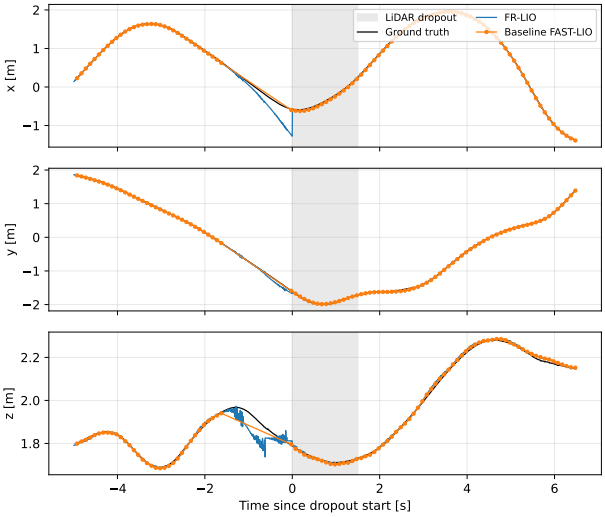


Figure 5: Position traces (x, y, z) before, during, and after a 1.5 s synthetic LiDAR dropout (gray band). The modified implementation (blue) continues publishing through the dropout via chained IMU propagation, accumulating ∼0.49 m of drift in body-frame position; baseline FAST-LIO (orange) freezes at the last EKF state until LiDAR resumes. Both implementations re-converge to ground truth (black) within one update after LiDAR returns.

axis. The largest reduction is on v_z, where the propagator's peak-to-peak noise drops from 6.6 m/s to 2.1 m/s, eliminating the high-bandwidth content that would otherwise excite the thrust loop. The magnitude of the unfiltered v_z noise reflects the operating environment of a body-mounted IMU on a quadrotor: motor harmonics in the 100–300 Hz range alias into the 200 Hz sample rate, and even a small misalignment of the estimated gravity vector projects onto v_z when integrated. The EMA at $f_c \approx 1.6$ Hz attenuates this content without touching the sub-Hz controller-bandwidth signal.

7.4 LiDAR-Interruption Resilience

To exercise the chained-propagation fallback, we apply the offline 1.5 s LiDAR-removal described above to the 60 s sequence used in Figure 4 and replay the resulting bag through both implementations. Figure 5 shows the resulting position traces. The modified implementation continues publishing throughout, accumulating approximately 49 cm of drift across the dropout. This drift reflects open-loop IMU integration at hover and converges back to ground truth on the next IESKF correction. The baseline implementation freezes at the last EKF state and resumes only when LiDAR returns. A controller running at 200 Hz under the modified implementation thus sees a smoothly drifting estimate rather than a discontinuity, which is the property that keeps the inner loop stable across the dropout.

8 Conclusion

We presented a set of architectural modifications that transform a benchmark-oriented tightly coupled LiDAR–inertial odometry implementation into a flight-ready system for embedded UAV platforms. Experimental results demonstrate that these changes substantially improve deployment characteristics without altering the underlying estimator architecture. As shown in Figure 3, the proposed system increases odometry output from the baseline scan-coupled rate of approximately 100 ms between updates (10 Hz) to a stable 5.0 ms interval (200 Hz), while significantly reducing timing variability. The modifications also preserve—and in some cases improve—estimation quality. Table 2 shows that trajectory accuracy remains comparable to the baseline in median ATE (3.8 cm vs. 4.7 cm), while reducing ATE RMSE from 11.7 cm to 5.2 cm and approximately halving the 1 s relative pose error (4.9 cm to 2.4 cm). Body-frame velocity estimation further achieves tracking errors below 0.1 m/s on all axes while attenuating high-frequency noise, particularly on the vertical channel where peak-to-peak fluctuations decrease from 6.6 m/s to 2.1 m/s (Figure 4).

Finally, Figure 5 demonstrates resilience to transient sensor failures: during a synthetic 1.5 s LiDAR interruption, the proposed approach continues publishing state estimates and limits drift accumulation to approximately 0.49 m, whereas the baseline freezes until measurements resume. Together, these results show that the proposed modifications improve timing, velocity availability, and robustness while preserving estimator accuracy. Because the IESKF + ikd-Tree estimation core remains unchanged, the approach can be directly

http://www.imavs.org/

integrated into FAST-LIO2-derived systems and generalized to other LIO implementations where correction updates operate more slowly than IMU propagation.

ACKNOWLEDGMENT

This work has been supported by the project SHEREC "Safe Healthy and Environmental Ship Recycling", Referece: 101136056, funded by European Union under the Horizon Europe Program HORIZON-CL4-2023-HUMAN-01 CNECT.

REFERENCES

[1] Wei Xu, Yixi Cai, Dongjiao He, Jiarong Lin, and Fu Zhang. FAST-LIO2: Fast direct LiDAR-inertial odometry. *IEEE Transactions on Robotics*, 38(4):2053–2073, August 2022.

[2] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5135–5142, 2020.

[3] Dongjiao He, Wei Xu, Nan Chen, Fanze Kong, Chongjian Yuan, and Fu Zhang. Point-LIO: Robust high-bandwidth light detection and ranging inertial odometry. *Advanced Intelligent Systems*, 5(7):2200459, 2023.

[4] Kenny Chen, Ryan Nemiroff, and Brett T. Lopez. Direct LiDAR-inertial odometry: Lightweight LIO with continuous-time motion correction. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3983–3989, 2023.

[5] 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.

[6] Patrick Geneva, Kevin Eckenhoff, Woosik Lee, Yulin Yang, and Guoquan Huang. OpenVINS: A research platform for visual-inertial estimation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4666–4672, 2020.

[7] Joan Solà. Quaternion kinematics for the error-state Kalman filter, 2017. arXiv:1711.02508.

[8] 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, 2024.

[9] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2520–2525, 2011.

http://www.imavs.org/

LiDARFlow: Real-Time Panel-Based MAV Guidance in Unknown Environments

João Machado[1]*, Zeynep Bilgin[2], Matthieu Verdoucq[1], Murat Bronz[1]
[1]ENAC, École Nationale de l'Aviation Civile, Université de Toulouse, Toulouse, France
[2]CNRS, Centre national de la recherche scientifique, France

http://www.imavs.org/

Abstract

This paper presents a guidance algorithm for micro aerial vehicles operating in unknown, cluttered environments using only onboard sensing. The method is based on a panel formulation originally derived from aerodynamic potential-flow theory and generates smooth, collision-free guidance vectors from locally perceived obstacles. The approach is extended to unknown environments by constructing and updating the obstacle representation online from onboard LiDAR measurements. The resulting obstacle-avoidance field is integrated with a nominal guiding vector field to produce the final control input. The system is experimentally validated in indoor flight tests under two scenarios: waypoint navigation and directional guidance. In both cases, the vehicle successfully completes its task while avoiding all obstacles in real time using only onboard perception. The results demonstrate that the method is computationally lightweight and suitable for onboard implementation, with point-cloud processing identified as the main practical limitation. These results support the feasibility of lightweight onboard guidance in unknown environments.

Keywords: micro aerial vehicles, autonomous navigation, obstacle avoidance, LiDAR, panel method, unknown environments.

1 Introduction

Advances in technology have made autonomous micro aerial vehicles (MAVs increasingly important across a wide range of applications such as search and rescue and disaster relief. Thanks to their small size and low weight, MAVs can reach locations inaccessible to humans, be deployed in large numbers, and provide vantage points that would otherwise be unattainable. In many of these scenarios, MAVs must operate in unknown environments for which no prior map is available. For example, in forests or disaster zones, pre-mapping all obstacles is often infeasible. Consequently, onboard sense-and-avoid capability is essential for mission success. Operating reliably under these conditions requires a robust guidance system that is lightweight enough to run onboard the MAV.

*Email address: joao.machado@enac.fr, zeynep.bilgin@irit.fr , matthieu.verdoucq@enac.fr, murat.bronz@enac.fr

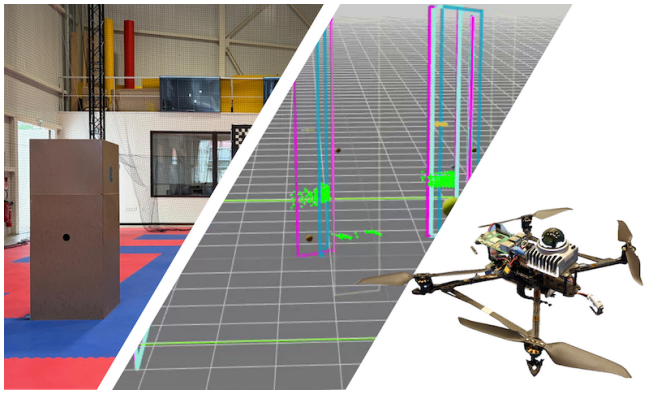


Figure 1: Stage overview of the project. The obstacles, represented on the left, are reconstructed with LiDAR data, as illustrated in the middle. The model of the drone used during the experiments can be seen on the write. It runs onboard with a Raspberry Pi running the nominal guidance and *PGFlow* algorithms, producing velocity commands that are passed to the onboard paparazzi micro-controller.

In many of these applications, MAVs are deployed by personnel who may not have extensive experience in piloting aerial robots. This motivates the development of guidance systems that reduce operator workload by abstracting low-level control tasks such as continuous obstacle avoidance, allowing users to focus on high-level mission objectives. Such capabilities are particularly valuable in time-critical missions where trained operators may be limited or unavailable.

1.1 Related work

Existing approaches to UAV guidance in unknown environments fall broadly into two categories: optimization-based methods and reactive methods. The first formulates guidance as a trajectory optimization problem and has converged on two competing paradigms. Hard-constraint corridor and mixed-integer (MIQP) planners such as FASTER [1], MADER [2], HDSM [3], DYNUS [4], and SANDO [5] guarantee collision-free trajectories. Whereas soft-constraint gradient-based planners like EGO-Planner [6], EGO-Swarm [7], and SUPER [8] embed collision risk as a penalty term to achieve faster computation at the cost of any formal safety guarantee. These methods typically pair a

http://www.imavs.org/

global planner with a local planner that compute and refine a safe path online. Their main drawbacks are that they generally rely on constructing and maintaining a map of the environment and that solving the underlying optimization is computationally demanding, which can limit the achievable replanning rate on the constrained hardware of a small MAVs.

The second category comprises reactive methods, which compute control commands directly from instantaneous sensor data. Recent reactive guidance for UAVs in cluttered, unknown environments has developed along three main lines. The first is control barrier function (CBF) safety filters operating directly on LiDAR or point-cloud data, exemplified by the composite CBF filter of Harms et al. [9], which steers an onboard LiDAR quadrotor through unknown indoor and outdoor clutter while handling thousands of constraints. The second is velocity obstacle and ORCA schemes, increasingly used together with model-predictive control [10] and reinforcement learning [11] for multi-agent and dynamic obstacle avoidance. The third is potential field, guidance vector field methods, which remain heavily used but are increasingly embedded as components within optimisation frameworks as local planners rather than deployed as standalone controllers. Because they act on live sensor measurements rather than on a global map, reactive methods offer low computational cost, short reaction latency, and map free operation, making them well suited to the limited onboard resources of an MAV. Their classical weaknesses lie in completeness, optimality, oscillation, and the absence of continuous time safety guarantees. Potential field methods in particular remain widely used but are well known to suffer from local minima, where competing attractive and repulsive forces can trap the vehicle short of its goal. In [12] this is mitigated by embedding the field within an MPC formulation .

The guidance method proposed in this work superposes the velocity vector fields generated by two algorithms. The nominal motion is generated using the convergent guiding vector field proposed in [13], while obstacle avoidance is achieved using the harmonic potential field approach of [14, 15]. Both algorithms are individually free of local minima under their respective assumptions, while being computationally light. Although no formal proof exists that their superposition preserves this property, [13] provides an intuitive argument for why undesired local minima are unlikely to arise. The practical effectiveness of the proposed guidance strategy is further demonstrated through the experimental results presented in section 4.

The harmonic potential field approach of [14] assumes complete *a priori* knowledge of obstacle locations and geometries. Although several studies have extended potential flow guidance to unknown environments, important limitations remain. In [16], obstacle avoidance was demonstrated for an aerial vehicle operating in an unknown environment, but only in hardware-in-the-loop simulation. The related panel method framework proposed in [17] was validated in unknown environments using a ground robot rather than an aerial platform. More recently, [18] considered potential flow guidance for UAVs in unknown environments, but validation was limited to simulation. This work addresses these limitations by proposing an onboard lightweight system that integrates LiDAR-based obstacle detection with a collision-free guidance strategy that operates without prior knowledge of the environment and is validated through real flight experiments.

1.2 Contributions

This paper presents a computationally lightweight onboard guidance system for MAV navigation in unknown environments using only onboard LiDAR sensing. The proposed approach combines a convergent guiding vector field with harmonic potential-flow obstacle avoidance and operates without prior knowledge of obstacle locations or geometries. The system is validated experimentally in two representative scenarios: waypoint navigation and commanded-direction flight.

The main contributions are:

- A guidance system that autonomously avoids obstacles, reducing operator workload by removing the need for manual obstacle avoidance;
- Full onboard integration of a computationally light functional LiDAR obstacle detection system and collision-free guidance algorithm.

An overview of the project can be seen in Fig. 1.

2 PROBLEM FORMULATION

The objective is to integrate LiDAR dynamic obstacle detection with a collision-free guidance algorithm. To validate the proposed system, two representative flight scenarios are considered in an unknown environment with unknown and dynamic obstacles. In both cases, the MAV has no prior map of the environment and must continuously detect and avoid obstacles in real time using onboard sensing.

The first scenario considers waypoint navigation. The MAV is initially located at an arbitrary position and is commanded by a human operator to reach a goal waypoint. Upon reaching this location, the operator may update the mission by assigning a new waypoint (e.g., returning to the starting location). The vehicle must therefore repeatedly track user-defined waypoints while autonomously handling all obstacle avoidance.

The second scenario considers directional guidance. Instead of fixed waypoints, the operator specifies a desired direction of motion that can be updated during flight. The MAV is required to continuously progress along the commanded direction while autonomously avoiding obstacles encountered along its path.

http://www.imavs.org/

Together, these two scenarios evaluate the system under operational modes where operators do not need extensive piloting expertise, relying on autonomous obstacle avoidance to safely execute missions in cluttered and unknown environments.

3 SYSTEM ARCHITECTURE

The guidance algorithm presented in [13] is used to generate the drone's nominal motion. In addition to its low computational cost, the algorithm is particularly well suited to the proposed approach because it is based on the superposition of vector fields. The original formulation does not address obstacle avoidance and so it is augmented with an obstacle-avoidance module described in the next section, which likewise relies on the superposition of velocity vector fields.

Obstacles are detected using an onboard Livox MID-360 LiDAR sensor, which produces dense three-dimensional point clouds which are processed into usable obstacles and sent to the guidance algorithm. The full system architecture can be observed in Fig. 2.

3.1 Nominal motion

To apply the algorithm in [13], it is necessary to compute the normal and tangent vectors to the point and line manifolds. The position of a point can be expressed in a three-dimensional Cartesian space by:

$$x - p_x = 0 \quad y - p_y = 0 \quad z - p_z = 0$$

where $(p_x, p_y, p_z) \in \mathbb{R}^3$. Defining $F_p(x,y,z) := \begin{bmatrix} F_{p,1} & F_{p,2} & F_{p,3} \end{bmatrix}^\intercal = \begin{bmatrix} x - p_x & y - p_y & z - p_z \end{bmatrix}^\intercal$, the normal and tangent manifold vectors can be obtained [13]:

$$\nabla F_{p,1} := \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^\intercal \quad \nabla F_{p,2} := \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^\intercal$$
$$\nabla F_{p,3} := \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\intercal \qquad \tau_p := \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\intercal$$

Accordingly, a line parallel to xOy can be defined as:

$$z - z_0 = 0$$
$$-\sin\alpha(x - x_0) + \cos\alpha(y - y_0) = 0$$

where $(x_0, y_0, z_0) \in \mathbb{R}^3$ is an offset given to the line and α the desired angle between the line and the x-axis. Defining $F_l(x,y,z) := \begin{bmatrix} F_{l,1} & F_{l,2} \end{bmatrix}^\intercal = \begin{bmatrix} z - z_0 & -\sin\alpha(x - x_0) + \cos\alpha(y - y_0) \end{bmatrix}^\intercal$, the normal and tangent manifold vectors can be obtained:

$$\nabla F_{l,1} := \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\intercal \qquad \nabla F_{l,2} := \begin{bmatrix} -\sin\alpha & \cos\alpha & 0 \end{bmatrix}^\intercal$$
$$\tau_l := \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \end{bmatrix}^\intercal$$

These can be directly introduced into the guidance algorithm to compute a velocity command prior to obstacle detection, V:

$$\begin{aligned} &\text{Point:} \quad V = -v_d \frac{k_c}{3} \sum_{i=1}^{3} F_{p,i} \frac{\nabla F_{p,i}}{\|\nabla F_{p,i}\|} \\ &\text{Line:} \quad V = v_d \left(\frac{\tau_l}{\|\tau_l\|} - \frac{k_c}{2} \sum_{i=1}^{2} F_{l,i} \frac{\nabla F_{l,i}}{\|\nabla F_{l,i}\|} \right) \end{aligned} \tag{1}$$

where v_d is the desired velocity along the tangent vector[1] and k_c is an arbitrary convergence gain. Note that for the line case, the drone is not converging to a particular line path. It is following the direction of the line, and so (1) reduces to:

$$\text{Line:} \quad V = v_d \frac{\tau_l}{\|\tau_l\|}$$

See [13] for a detailed derivation of the algorithm.

3.2 Obstacle avoidance with PGFlow

For obstacle avoidance, the PGFlow algorithm [15, 19] is employed. PGFlow is a fluid dynamics inspired method that models obstacles as collections of one dimensional panels in a two-dimensional plane. Each panel is represented by a line source (or vortex, depending on the formulation), generating an irrotational velocity field around the obstacle. Fig. 3 illustrates how an obstacle is decomposed into panels.

Since obstacle avoidance is generally more critical in the horizontal plane, only the x and y components of the repulsive field are considered. Although the method can be extended to three dimensions, this significantly increases computational complexity. Vertical collision avoidance can instead be handled through a simpler repulsion mechanism acting along the z-axis.

Given a panel i with vertices $p_{oi_1} \to (x_{oi_1}, y_{oi_1})$ and $p_{oi_2} \to (x_{oi_2}, y_{oi_2})$, its control point is defined as the centre point of the panel.

$$p_{ci} := \frac{(p_{oi_2} - p_{oi_1})}{2} + p_{oi_1} \to (x_{ci}, y_{c_i})$$

The velocity command given to the drone is:

$$\dot{x} = V_x - \sum_{i=1}^{m} \frac{\lambda_i}{2\pi} \int_i \frac{\partial}{\partial x} \ln R_i \mathrm{d}l_i$$
$$\dot{y} = V_y - \sum_{i=1}^{m} \frac{\lambda_i}{2\pi} \int_i \frac{\partial}{\partial y} \ln R_i \mathrm{d}l_i$$

where $V = \begin{bmatrix} V_x & V_y \end{bmatrix}^\intercal$ is the velocity command prior to *PGFlow*, m is the total number of panels, λ_i is the strength of panel i, R_i is the distance from the drone to a point on panel i, and $\mathrm{d}l_i$ is the differential length element along panel i.

The panel strengths are computed such that the normal velocity at each panel control point satisfies $V_i \geq 0$, preventing trajectories from penetrating the obstacle boundary and thereby producing a repulsive effect:

[1] for the point case, since the tangent vector is $\mathbf{0}$, this can be simply treated as a gain and the total convergence gain becomes $k_c' := v_d k_c$.

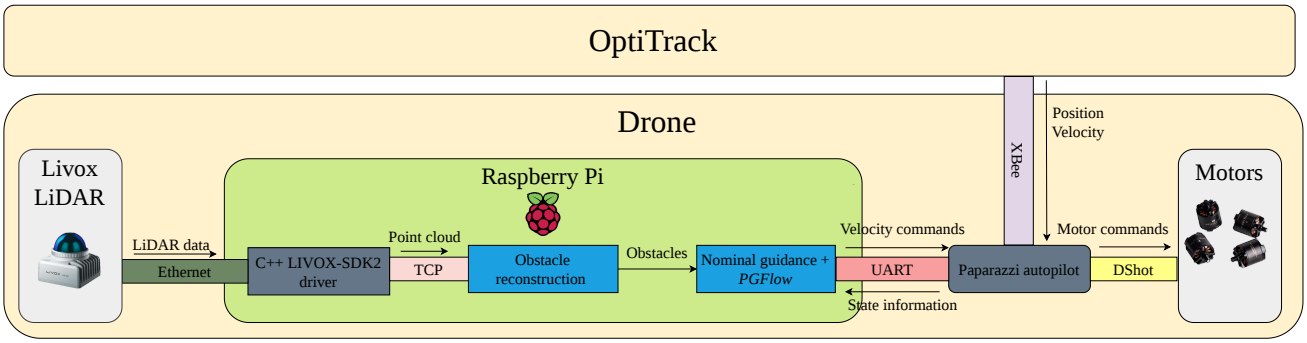


Figure 2: System architecture and respective data flow. LiDAR sends captured data to the Raspberry Pi where it is then processed by the different modules. The guidance algorithm generates the velocity commands sent to the autopilot. The autopilot fuses the vehicle's position and velocity captured by OptiTrack into the full vehicle's state, sending it to the Raspberry Pi. It also convert the velocity commands into the appropriate motor commands.

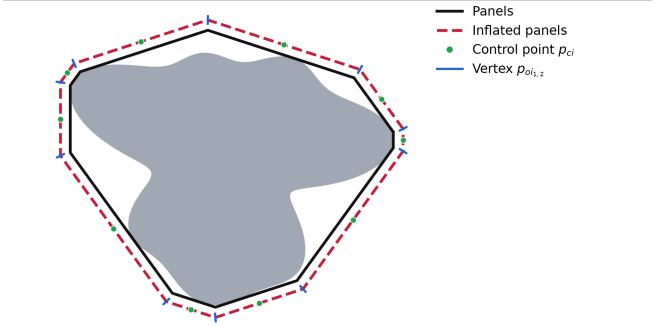


Figure 3: The obstacle (gray) is decomposed into panels (black) that act as a line source for the guidance of the drone. The panels are inflated (red) by at least half the drone's size so that collision avoidance is guaranteed. The control points (green) are located at each panel's half length.

For panel i :

$$\frac{\lambda_i}{2} + \sum_{i \neq j}^{m} \frac{\lambda_j}{2\pi} I_{ij} = -V_i + V(x_{ci}, y_{ci}) \cdot \overrightarrow{n_i}$$

where I_{ij} is the panel interaction matrix [15], and $\mathbf{n}_i$ is the outward-pointing normal vector of panel i. See [15] and [14] for a detailed derivation and discussion of the algorithm.

To account for the physical dimensions of the drone, obstacles must be inflated by at least half the characteristic size of the vehicle. Under these conditions, the method guarantees that the drone does not collide with the obstacles. This is also validated empirically on section 4.

In the proposed implementation, the obstacle panels used by PGFlow are not known a priori. They are extracted from onboard LiDAR data.

3.3 *LiDAR-based obstacle extraction*

Obstacle detection is performed using a Livox MID-360 solid-state LiDAR operating at approximately 10 Hz. At each scan, the sensor produces a high-density point cloud in the LiDAR local coordinate frame. The objective of the perception layer is to extract a compact set of obstacles that can be directly used by PGFlow.

The LiDAR scans are acquired onboard by a Raspberry Pi using a C++ driver based on Livox-SDK2 and published through a ZeroMQ stream. The drone's position is provided by the OptiTrack via XBee radio modem fused onboard with the drone's IMU. The LiDAR scans are then associated with the corresponding state estimate and transformed into a common ENU frame.

To reduce computational cost, the point cloud is first downsampled using a voxel grid with voxel side length $s = 0.2$ meters. Each point $p_i = (x_i, y_i, z_i)$ is assigned to a voxel index

$$K_i = 1000003\, k_{x_i} + 1001\, k_{y_i} + k_{z_i},$$

where

$$k_{x_i} = \left\lfloor \frac{x_i}{s} \right\rfloor, \qquad k_{y_i} = \left\lfloor \frac{y_i}{s} \right\rfloor, \qquad k_{z_i} = \left\lfloor \frac{z_i}{s} \right\rfloor.$$

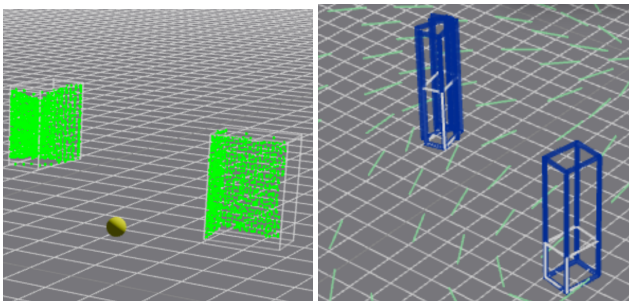
All points assigned to the same voxel are replaced by their centroid. The resulting cloud is then cropped to the region of interest, which corresponds to the size of the used portion of the arena,

$$x, y \in [-3, 3]\ \mathrm{m}, \qquad z \in [0.5, 3]\ \mathrm{m},$$

and points closer than 0.2 m to the sensor are discarded to remove returns from the vehicle body or from very near-field artifacts.

The filtered points are projected onto the horizontal plane and grouped into clusters. Candidate obstacle faces are extracted from each cluster using iterative RANSAC line fitting, allowing multiple planar faces to be identified within a single cluster. The extracted line segments are filtered according to geometric consistency, including minimum length, point support, continuity, and thickness, to reject spurious detections and over-extended fits.

http://www.imavs.org/



(a) (b)

Figure 4: Raw pointcloud generated by the LiDAR (a), in green, and resulting obstacle segments, in blue, after processing by RANSAC (b).

The remaining segments are associated with a temporal segment map, where compatible observations update existing segments and unmatched observations initialize new ones. Repeated detections increase the confidence of a segment, while temporary occlusions are handled by maintaining segments for a limited duration before removal.

The perception layer outputs an obstacle-indexed collection of segments, each defined by its endpoints, tangent direction, outward normal, and confidence. This compact representation is passed directly to the PGFlow guidance algorithm for real-time obstacle avoidance. The raw LiDAR generated pointcloud can be observed in Fig. 4a. Fig. 4b shows the obstacle segments generated from that same pointcloud. The full LiDAR obstacle detection and processing pipeline can be observed in Fig. 5.

4 RESULTS

4.1 Experimental setup

Indoor experiments are conducted in the Toulouse Occitanie Drone Flight Arena at ENAC, Toulouse, France. The flight arena measures 10×10 meters, and all experiments are performed using a quadrotor equipped with the Paparazzi autopilot system [20]. The arena is equipped with a 16-camera OptiTrack motion capture system, which provides a highly accurate real-time state estimation of the vehicle.

During flight, the vehicle's position and velocity are estimated by the OptiTrack system and transmitted via XBee radio modem and fused onboard with Paparazzi's IMU for full state estimation. Obstacle perception is performed exclusively onboard using the LiDAR sensor described in the previous section.

The obstacles are placed at fixed locations in the arena, as shown in Fig. 6. However, their positions are not known *a priori* to the drone and are assumed to be non-static from the vehicle's perspective. Consequently, the drone continuously scans the environment and executes the obstacle detection algorithm at every time step.

The physical specifications of the vehicle can be found in Table 1.

Table 1: Physical specifications of the vehicle used for the experiments.

Specification	Value
Mass	1200 g
Maximum thrust	40 N
Battery	7000 mAh, 11.1 V (3-cell)
Motors	T-motor air gear 450 V2
Flight time	$\sim$20 min
LiDAR	Livox Mid-360, 10 Hz
Companion computer	Raspberry Pi 5
Autopilot	Tawaki V2.0 (Paparazzi)

4.2 Scenario 1: Waypoint guidance

In the first scenario, the vehicle is required to navigate from waypoint $A = (-2, -4)$ meters to waypoint $B = (2, 4)$ meters and return to the starting position. The chosen parameters for (1) are $v_d = 0.5$ meters per second and $k_c = 1$. The maximum velocity is limited to 0.5 meters per second for safety. Two obstacles are placed at $(-1.5, -1.75)$ meters and $(-0.5, 2)$ meters.

The resulting trajectory is shown in Fig. 7a. The vehicle successfully reaches both waypoints while maintaining collision-free operation throughout the experiment.

4.3 Scenario 2: Directional guidance

In the second scenario, the vehicle is tasked with following a reference direction. The commanded direction is initially $(0, 1)$ and is later changed to $(1, 0)$. The chosen parameters for (1) are $v_d = 0.5$ meters per second and $k_c = 1$. The maximum velocity is again limited to 0.5 meters per second. Three obstacles are placed at $(-2, 1)$ meters, $(1.5, 0.5)$ meters, and $(-1, -2)$ meters.

The resulting trajectory is shown in Fig. 7b. The vehicle is able to track the commanded direction while avoiding the obstacles.

5 DISCUSSION

The experimental results demonstrate the feasibility of the proposed guidance system in unknown environments using only onboard LiDAR sensing. In both validation scenarios, the MAV successfully achieved its objectives, reaching the requested waypoints in the waypoint-navigation scenario and following the commanded direction in the directional-guidance scenario while avoiding all detected obstacles. However, the resulting trajectories exhibit noticeable oscillations and sharp turns rather than smooth paths. This behaviour is primarily attributed to instabilities in the obstacle detection algorithm. The estimated obstacles changed dimensions erratically between frames. These fluctuations caused corresponding variations in the repulsive forces generated by *PGFlow*, leading to oscillatory trajectory corrections.

The MAV operated at a relatively low speed of approximately $0.5, \mathrm{m/s}$. Further validation is required before evaluating the proposed guidance system at higher flight speeds.

http://www.imavs.org/

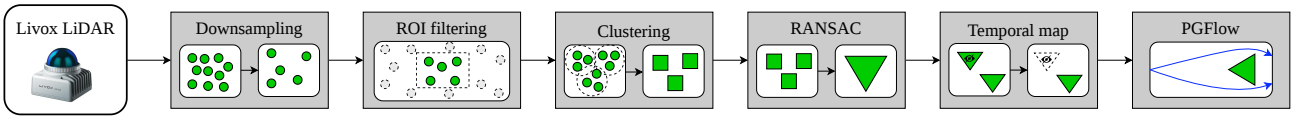
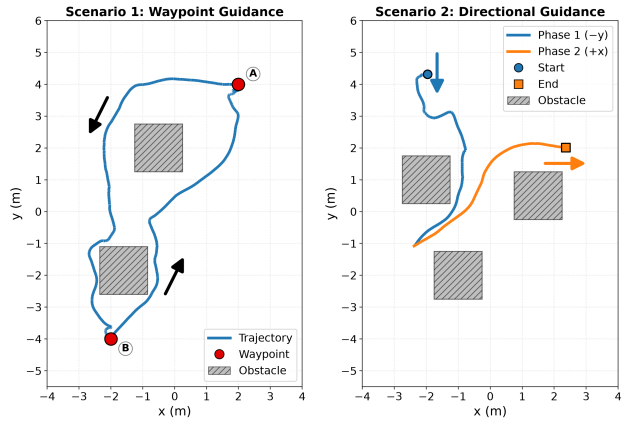


Figure 5: LiDAR's obstacle detection and processing pipeline. The raw data is captured by the Livox LiDAR sensor and then processed along the pipeline until it can be used by *PGFlow* to generate the appropriate velocity commands for the drone.



(a) (b)

Figure 6: Experimental setups for the go to waypoint, (a), and follow direction, (b) scenarios. On (a) two obstacles are placed between the waypoints the drone has to reach. On (b) the drone has to navigate around three obstacles while following the nominal direction command.



(a) (b)

Figure 7: Experimental drone trajectories for the first, (a), and second, (b), scenarios. (a) The drone starts at waypoint A, flies to waypoint B, and returns to waypoint A while moving counter-clockwise (black arrows). The trajectory (blue) successfully avoids the obstacles (zebra-patterned squares). (b) For each coloured trajectory segment, the commanded direction is indicated by the arrow of the corresponding colour. The resulting trajectory successfully avoids the obstacles.

Additionally, all obstacles considered in the experiments were static. The current approach is not designed to handle fast-moving obstacles, as it relies solely on their instantaneous positions. To address this limitation, a future extension could incorporate an obstacle motion prediction module to estimate future obstacle positions.

Rather than continuously performing low-level obstacle avoidance, the operator is only required to specify a desired direction of travel while the vehicle autonomously reacts to the surrounding environment. Although no formal user study was conducted, this mode of operation has the potential to reduce operator workload and the level of piloting expertise required, making the approach attractive for applications such as search and rescue, where personnel may not have extensive experience operating MAVs.

The proposed guidance algorithm is computationally efficient, provided that the number of obstacle panels remains limited. At each obstacle update, an $n \times n$ matrix inversion is required, resulting in a computational complexity of $O(n^3)$, where n denotes the total number of panels. Consequently, the computational cost increases rapidly as additional panels are introduced. In practice, however, the number of panels can be kept small by removing redundant or distant panels while preserving an adequate approximation of the obstacle geometry. The *PGFlow* algorithm runs comfortably at 100 Hertz. As the number of panels increases, the maximum frequency at which the obstacle matrix can be computed decreases. The relationship between this frequency and the maximum allowed number of detected panels is summarized in Table 2. Even at higher frequencies the maximum allowed number of detected panels remains sufficiently high for reliable real-time obstacle avoidance.

Table 2: Relation between the frequency at which the obstacle detection algorithm is run and the maximum number of detected panels allowed for real-time safe obstacle avoidance.

Frequency (Hz)	1	5	10	20
Maximum number of panels	2025	1100	725	506

Obstacle perception proved to be one of the most challenging components of the system. The dense LiDAR point cloud makes it difficult to segment individual obstacles and group the corresponding points, resulting in an unstable obstacle estimation algorithm, where object dimensions would

http://www.imavs.org/

change erratically between frames. Furthermore, the complete point cloud is currently transmitted from the C++ LiDAR driver to the Python guidance module via TCP before any processing is performed. A more efficient implementation would perform the filtering and obstacle extraction directly within the C++ driver, reducing both communication overhead and computational cost. Machine-learning-based methods would probably be a better fit to convert the point cloud into concrete obstacles.

Several simplifying assumptions were made in this work. The vehicle state was obtained from an external OptiTrack motion capture system, eliminating the need for onboard state estimation. Furthermore, guidance was only considered in the horizontal plane, with motion along the z-axis being neglected. Consequently, the experimental validation focuses solely on the proposed guidance strategy.

Only convex obstacles were considered. Concave obstacles are more challenging to navigate around and may introduce stagnation points (local minima) in the guidance field. In practice, concave obstacles can be approximated using multiple convex panels, albeit at the cost of approximation accuracy.

Finally, the proposed system was validated using a single MAV. Extending the approach to multi-vehicle operation is a natural continuation of this work.

6 CONCLUSION

This paper presented a fully onboard integrated guidance system for MAV navigation in unknown environments, addressing two scenarios: convergence to user-defined waypoints and tracking of a commanded direction. Obstacle detection is performed using an onboard LiDAR sensor, while obstacle avoidance is ensured by a guidance algorithm [14, 15]. The nominal motion is generated using a guiding velocity vector field approach [13].

The system was experimentally validated in an indoor environment with unknown convex obstacles. In both scenarios, the MAV successfully achieved its objectives without encountering local minima, while maintaining low computational cost. It has not been tested with concave obstacles.

These results demonstrate the feasibility of fully onboard guidance and obstacle avoidance in unknown environments using only onboard sensing. This enables intuitive high-level safe control with minimal operator expertise.

REFERENCES

[1] Jesus Tordesillas, Brett T. Lopez, Michael Everett, and Jonathan P. How. FASTER: Fast and safe trajectory planner for navigation in unknown environments. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019. Journal version: IEEE Trans. Robotics, vol. 38, 2022, DOI 10.1109/TRO.2021.3100142.

[2] Jesus Tordesillas and Jonathan P. How. MADER: Trajectory planner in multiagent and dynamic environments. *IEEE Transactions on Robotics*, 38(1), 2022.

[3] Charbel Toumieh and Dario Floreano. High-speed motion planning for aerial swarms in unknown and cluttered environments. *IEEE Transactions on Robotics*, 2024.

[4] Kota Kondo, Mason Peterson, Nicholas Rober, Juan Rached Viso, Lucas Jia, Jialin Chen, Harvey Merton, and Jonathan P. How. DYNUS: Uncertainty-aware trajectory planner in dynamic unknown environments, 2025. arXiv preprint.

[5] Kota Kondo, Jesús Tordesillas, and Jonathan P. How. SANDO: Safe autonomous trajectory planning for dynamic unknown environments, 2026. arXiv preprint.

[6] Xin Zhou, Zhepei Wang, Chao Xu, and Fei Gao. EGO-planner: An ESDF-free gradient-based local planner for quadrotors. *IEEE Robotics and Automation Letters*, 6(2), 2021.

[7] Xin Zhou, Jiangchao Zhu, Hongyu Zhou, Chao Xu, and Fei Gao. EGO-swarm: A fully autonomous and decentralized quadrotor swarm system in cluttered environments. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[8] Yunfan Ren, Fangcheng Zhu, Guozheng Lu, Yixi Cai, Longji Yin, Fanze Kong, Jiarong Lin, Nan Chen, and Fu Zhang. SUPER: Safety-assured high-speed navigation for MAVs. *Science Robotics*, 10(98):eado6187, 2025.

[9] Marvin Harms, Martin Jacquet, and Kostas Alexis. Safe quadrotor navigation using composite control barrier functions. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6343–6349, 2025.

[10] Jihao Huang, Xuemin Chi, Jun Zeng, Zhitao Liu, and Hongye Su. GeoPro-VO: Dynamic obstacle avoidance with geometric projector based on velocity obstacle, 2024. arXiv preprint.

[11] Gang Zhong, Yupu Liu, Sen Du, Fei Wang, Jinlun Zhou, and Honghai Zhang. 3D RVO-enhanced multi-agent deep reinforcement learning for collision avoidance in urban structured airspace. *Aerospace Science and Technology*, 2025.

[12] Nikita Vaibhav Pavle, Shrreya Rajneesh, Rakesh Kumar Sahoo, and Manoranjan Sinha. Obstacle avoidance of UAV in dynamic environments using direction and velocity-adaptive artificial potential field, 2025. arXiv preprint.

http://www.imavs.org/

[13] João Machado, Matthieu Verdoucq, Christophe Jouffrais, and Murat Bronz. Swarm choreography made simple: Superposed guiding vector fields for rigid formation path following. In *14th International Conference on Unmanned Aircraft Systems (ICUAS)*, Corfu, Greece, 2026.

[14] Zeynep Bilgin, Ilkay Yavrucuk, and Murat Bronz. Urban air mobility guidance with panel method: Experimental evaluation under wind disturbances. *Journal of Guidance, Control, and Dynamics*, 2024.

[15] Farbod Fahimi. Obstacle avoidance using harmonic potential functions. In Farbod Fahimi, editor, *Autonomous Robots: Modeling, Path Planning, and Control*, pages 1–49. Springer US, 2009.

[16] G. C. S. Cruz and P. M. M. Encarnação. Obstacle avoidance for unmanned aerial vehicles. *Journal of Intelligent & Robotic Systems*, 65(1–4):203–217, 2012.

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

[18] J. Rudzki, Zeynep Bilgin, and Ilkay Yavrucuk. Enhancing unmanned rotorcraft guidance with LIDAR and ADS-B integrated PGFlow algorithm. In *50th European Rotorcraft Forum (ERF)*, Germany, 2024.

[19] Zeynep Bilgin, Ilkay Yavrucuk, and Murat Bronz. Urban air mobility guidance with panel method: Experimental evaluation under wind disturbances. *Journal of Guidance, Control, and Dynamics*, 47(6):1080–1096, 2024.

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

http://www.imavs.org/

Monocular Geo-Localization of Terrestrial Objects from UAV Imagery with Spatial Refinement

Heitor Riquelme[1], João G. Melo[1], João P. Mafalado[1], Pedro César[1], Driele Pires[1], Mateus Ferreira[1], Bruno Eduardo[2], Edna Barros[1] and Paulo Rosa[2]
[1]Centro de Informática, Universidade Federal de Pernambuco,
Av. Prof. Moraes Rego, 1235 - Cidade Universitária, Recife - Pernambuco, Brazil.
[2]Department of Computer Engineering, Instituto Militar de Engenharia,
Praça Gen. Tibúrcio, 80 - Urca, Rio de Janeiro - Rio de Janeiro, Brazil.

http://www.imavs.org/

ABSTRACT

Geo-localization of ground objects from UAV platforms is a fundamental capability for applications such as autonomous inspection, search and rescue, and target tracking. This paper presents a fully online pipeline for estimating the geographic coordinates of multiple terrestrial objects from UAV monocular imagery. The approach integrates (i) a timestamp interpolation scheme that synchronizes camera frames with asynchronous UAV state estimates, (ii) a geometric projection model that maps image detections to ground-plane coordinates using UAV attitude and position, and (iii) a spatial tracking stage based on the Hungarian algorithm and Kalman filtering that associates and refines repeated noisy estimates into a consistent per-object geo-location. Experiments conducted in simulation with Gazebo and ArduPilot SITL across 3 scenarios and 6 flight paths demonstrate a mean localization RMSE of 0.603 m, with 97.9% of targets successfully identified.

Keywords: Geo-localization; Multi-Object Tracking; Spatial Tracking; State Interpolation

1 INTRODUCTION

Geolocation of objects from aerial platforms is a key capability for applications such as mapping, inspection, search and rescue [1]. When using monocular cameras, this task is challenging due to the lack of depth information and sensor asynchrony between the imaging and navigation systems.

An additional challenge lies in maintaining consistent associations between detections over time. As a UAV traverses its trajectory, each physical target is observed multiple times from different viewpoints and under varying localization noise, producing geo-localization estimates that are spatially dispersed around the true location [2]. Furthermore, false detections may contaminate this set with spurious estimates. Consequently, these raw observations must be aggregated, filtered, and associated across frames to yield a consistent and accurate representation of the scene. To this end, this paper presents a method for estimating the geographic coordinates of multiple ground objects detected in UAV imagery, addressing the inherent noise and association ambiguity through a spatial refinement pipeline.

The main contributions of this work are:

- A timestamp interpolation method that synchronizes monocular camera frames with asynchronous UAV state estimates.
- A projection model for monocular geo-localization of ground objects from UAV attitude and position.
- A spatial tracking and refinement pipeline that associates, filters, and consolidates repeated detections into consistent per-object geo-locations.

2 SYSTEM OVERVIEW

2.1 Problem Statement

The objective of this work is to estimate the geographic locations of multiple terrestrial objects using a monocular camera onboard a UAV. During flight, the UAV captures images while its onboard Flight Controller Unit (FCU) provides position and attitude estimates. Objects of interest are detected in the images, and their geo-locations are estimated assuming they lie on the ground plane.

A fundamental difficulty arises from the fact that each physical object is typically observed multiple times from different viewpoints along the UAV trajectory. These repeated observations produce multiple geo-localization estimates per object, as illustrated in Fig. 1. Due to sensor noise, detection uncertainty, and projection errors, these estimates are spatially dispersed around the true object location. Additionally, false detections may introduce spurious estimates that do not correspond to any real object.

Consequently, the raw set of geo-localization estimates does not directly provide a reliable representation of the scene. Instead, it consists of a mixture of:

- Multiple estimates corresponding to the same object.
- Estimates belonging to different objects.
- Outliers caused by incorrect detections or localization errors.

Therefore, the problem consists not only of estimating object locations, but also of maintaining consistent associations between observations over time, correctly identifying the number of distinct objects, and assigning a single, accurate geo-location to each of them.

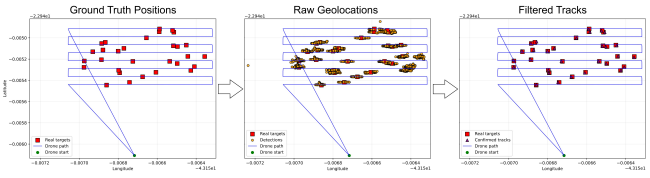


Figure 1: Illustration of the problem: multiple observations generate noisy geo-localizations that must be associated and refined into a single estimate per object.

2.2 *Overview of the Proposed Approach*

To address this problem, we propose the workflow illustrated in Figure 2, a fully online pipeline composed of two main components:

- **Object Detection and Position Estimation**
- **Spatial Refinement with Tracking and Filtering**

The first component processes each incoming frame independently. Objects are detected as bounding boxes in the image using a deep learning-based object detector; then, the UAV position and attitude are interpolated at the frame timestamp, and each detection is projected into geographic coordinates using geometric projection.

The second component maintains a set of object tracks over time. New geo-localization estimates are associated with existing tracks using a data association method based on the Hungarian algorithm [3]. Tracks are progressively refined, filtered, and either confirmed or discarded based on their temporal and spatial consistencies.

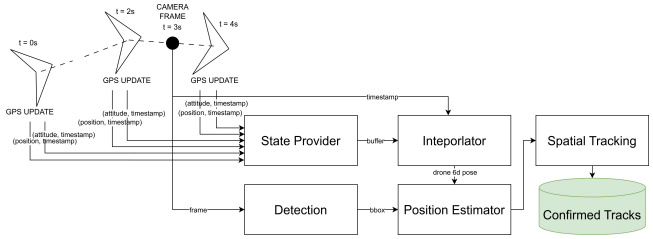


Figure 2: Workflow of the proposed approach.

3 GEO-LOCALIZATION

3.1 *Object Detection*

For each frame, objects are detected and represented as bounding boxes. The center of each bounding box is used as the representative image point. The timestamp at which each frame was captured is also recorded, enabling interpolation and estimation of the UAV's position and attitude at that moment.

We adopt a YOLOv11 model for object detection [4], owing to its suitability in real-time systems with high-throughput with low-power constraints [5, 6]. For each detected object, we record the bounding box center, capture timestamp, detection confidence, and object class. This information can be used further to improve multi-object tracking and association.

3.2 *State Provider and Interpolation*

We employ the ArduPilot's Software In The Loop (SITL) for UAV control and state estimation. By default, it provides estimations of the current position and attitude at 10 Hz. However, position and attitude data are published in separate messages, so they need to be read asynchronously. The state interpolator maintains two timestamped buffers, one for position and one for attitude, defined as:

$$(\mathbf{p}_i, t_i^p) \tag{1}$$

$$(\mathbf{a}_i, t_i^a) \tag{2}$$

where $\mathbf{p} = (x, y, z)$ denotes position in local ENU coordinates, $\mathbf{a} = (\psi, \theta, \phi)$ denotes the attitude as yaw, pitch, and roll angles, and t_i is the timestamp of the i-th measurement.

These position and attitude messages are saved in buffers in the State Provider, with their respective timestamps. This way, we keep track of the UAV's movements for up to a configurable number of messages, currently configured as $buffer\_size = 200$.

The interpolations are implemented as linear movements between timestamps, i.e., given a moment in time t, we compute a weighted average of the measurements taken at the neighboring timestamps. Position is interpolated linearly between consecutive measurements:

$$\mathbf{p}(t) = (1 - \alpha)\mathbf{p}_i + \alpha\mathbf{p}_{i+1} \tag{3}$$

where α is determined from timestamps, by:

$$\alpha = \frac{t - t_i}{t_{i+1} - t_i} \tag{4}$$

where $t_i \leq t \leq t_{i+1}$.

Attitude angles are circular quantities, so direct linear interpolation would produce incorrect results near discontinuities (e.g., wrapping from 359° to 1°). Instead, each angle is interpolated by averaging its sine and cosine components separately, then recovering the result via arctangent. Equation (5) shows how this applies to the yaw angle ψ:

$$\psi(t) = arctan\left(\frac{(1-\alpha)sin(\psi_i) + \alpha sin(\psi_{i+1})}{(1-\alpha)cos(\psi_i) + \alpha cos(\psi_{i+1})}\right) \tag{5}$$

http://www.imavs.org/

3.3 Position Estimator

This module takes as inputs the coordinates of a given pixel (u, v), the camera's intrinsic and extrinsic parameters, and the UAV's position and attitude at the time each frame was captured, as illustrated in Figure 3. Using a geometric projection model, it computes the geo-localization of the given pixel, which should correspond to the position of a desired object.

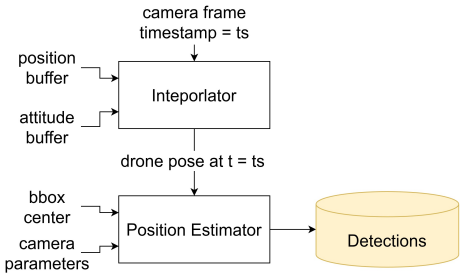


Figure 3: Inputs and outputs of the position estimator.

The geo-localization approach extends the method described in [7] to accommodate a lower-cost UAV platform without gimbal stabilization. In this setup, the camera is rigidly mounted and permanently oriented towards the ground. Therefore, the camera orientation is directly derived from the UAV's attitude using a fixed offset:

$$\begin{aligned} pitch_{cam} &= pitch - 90^\circ \\ yaw_{cam} &= yaw \\ roll_{cam} &= roll \end{aligned} \tag{6}$$

Since the camera is rigidly fixed to the UAV body without stabilization, any change in UAV attitude directly perturbs the camera pose and may introduce additional geo-localization error. This trade-off is accepted in exchange for a simpler, lower-cost hardware configuration compatible with custom-built platforms.

The method relies on three coordinate systems: the world coordinate system, the UAV coordinate system, and the camera coordinate system. The world coordinate system is defined in a metric georeferenced frame, such as UTM (Universal Transverse Mercator) or a local ENU (East-North-Up) frame, allowing positions to be expressed in linear units (meters).

In this formulation, all coordinate systems share the same origin, defined in the world reference frame. The UAV and camera coordinate systems differ from the world system only by their orientations:

- World coordinate system: $(x, y, z) = (\text{east}, \text{north}, \text{up})$
- Drone coordinate system: $(x, y, z) = (\text{forward}, \text{right}, \text{down})$
- Camera coordinate system: $(x, y, z) = (\text{right}, \text{down}, \text{forward})$

The relationship between these coordinate systems is defined by the camera pose. The rotation matrix R that maps points from the world coordinate system to the camera coordinate system is computed following the method described in [7]. This matrix encodes the camera orientation as derived from the UAV attitude and the fixed mounting configuration.

The camera center expressed in the world coordinate system is given by $v_0 = -R^T t$. Therefore, the translation vector t does not need to be computed independently, as the camera position is directly obtained from the UAV position, estimated by the state interpolator.

Given the camera pose, the projection from 3D world points to image pixels is modeled using the pinhole camera model [8]:

$$K[R \mid t]P = \lambda p, \tag{7}$$

where K is the intrinsic matrix, R (rotation matrix) and t (translation vector) map points from the world coordinate system to the camera coordinate system, $p = (u, v, 1)^T$ is the homogeneous pixel coordinate, λ is a scale factor, and $P = (x, y, z, 1)^T$ is a 3D point expressed in homogeneous coordinates.

To perform geo-localization, the inverse mapping must be computed. By rearranging the projection Equation (7), it is possible to derive the 3D ray in the world coordinate system corresponding to a given image pixel. This ray can be written in parametric form as:

$$P(\lambda) = v_0 + \lambda v, \tag{8}$$

where:

$$v = R^T K^{-1} p. \tag{9}$$

Thus, each pixel defines a ray originating from the camera center and extending into the scene.

The geo-location is obtained by computing the intersection between this ray and the ground. Assuming a planar ground model, this intersection can be obtained analytically. Let Δh denote the height difference between the camera center and the ground plane, and let v_z be the vertical component of the direction vector v. The intersection point is then given by:

$$P_{ground} = v_0 - \frac{\Delta h}{v_z} v. \tag{10}$$

This formulation provides a direct and efficient way to map image pixels to 3D positions in the world coordinate system under the planar ground assumption.

4 SPATIAL REFINEMENT

The purpose of this stage is to maintain a consistent set of object tracks over time from noisy geo-localization estimates.

http://www.imavs.org/

The tracker maintains and updates this set of object representations upon the arrival of new geo-localization estimates at each frame. The overall process is illustrated in Fig. 4.

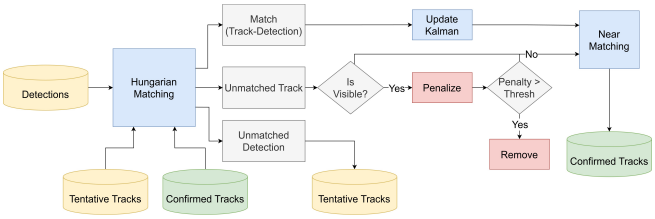


Figure 4: Overview of the spatial refinement pipeline. New detections are associated with existing tracks using the Hungarian algorithm. Tracks are updated, penalized, or removed depending on their consistency over time.

Each track represents a candidate object and is defined by:

- Estimated geographic position (latitude and longitude).
- State covariance (for uncertainty modeling).
- Track status (tentative or confirmed).
- A penalty counter used for track management.

Two sets of tracks are maintained:

- **Tentative Tracks**: recently initialized tracks that require validation.
- **Confirmed Tracks**: stable tracks corresponding to persistent objects.

At each time step, a set of new geo-localization estimates (detections) is obtained. These detections are associated with existing tracks (both tentative and confirmed) using the Hungarian algorithm.

A cost matrix is constructed based on the Mahalanobis distance between detections and track estimates:

$$d_M(x,y) = \sqrt{(\mathbf{x}_i^{det} - \mathbf{y}_j^{track})^T S^{-1} (\mathbf{x}_i^{det} - \mathbf{y}_j^{track})} \quad (11)$$

where $\mathbf{y}_i^{track}$ and $\mathbf{S}$ are, respectively, the mean and the covariance matrix of track i, and $\mathbf{x}_j^{det}$ is the position of detection j.

Assignments are accepted only if the distance is below a predefined gating threshold. The association step produces three sets:

- Matched track-detection pairs.
- Unmatched tracks.
- Unmatched detections.

4.1 *Track Update with Kalman Filtering*

For each matched track-detection pair, the track state is updated using a Kalman filter. The detection is treated as a measurement of the track position, allowing the filter to reduce noise and improve temporal consistency.

The update step refines both the estimated position and its associated uncertainty. This enables smoother trajectories and more robust associations in subsequent frames.

4.2 *Handling Unmatched Detections*

Detections that are not associated with any existing track are used to initialize new tentative tracks. These tracks represent potential new objects but require further validation before being confirmed.

4.3 *Handling Unmatched Tracks*

Tracks that are not matched with any detection are processed based on their visibility:

- If the track is expected to be visible in the current frame but is not detected, it is penalized by increasing its penalty counter.
- If the penalty exceeds a predefined threshold, the track is removed, as it is considered unreliable.

Visibility is determined by the camera's field of view and the estimated track position. This step helps to remove outliers from the set.

4.4 *Near Matching and Duplicate Removal*

To further improve consistency, a near-matching heuristic is applied to the set of confirmed tracks. Since repeated observations of the same physical object may occasionally generate multiple independent trajectories, tracks that are spatially close to one another are assumed to correspond to the same target and are therefore merged into a single representation.

This heuristic helps reduce duplicate detections and contributes to maintaining a compact and spatially consistent representation of the environment. However, its effectiveness depends on the selected distance threshold, particularly in scenarios where multiple targets are positioned very close to each other.

4.5 *Final Output*

At any time step, the system maintains a set of confirmed tracks, each associated with a distinct physical object and identified by a unique ID. The geographic coordinates associated with these tracks are used as the system's final output.

5 EXPERIMENTS

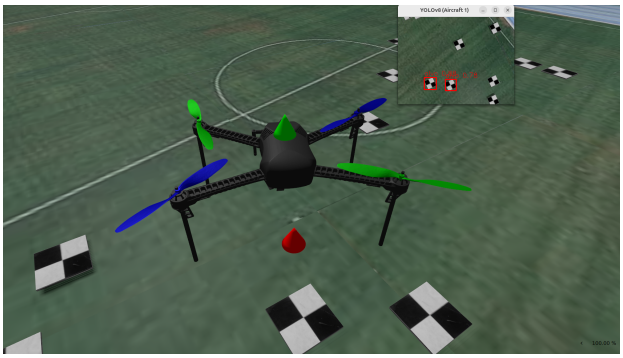
5.1 *Experimental Setup*

Experiments were conducted in a simulated environment using Gazebo, with ArduPilot's SITL providing UAV control and state estimation. The UAV was equipped with a downward-facing monocular camera; frames were published to a ROS 2 topic and streamed via RTSP using GStreamer.

http://www.imavs.org/

http://www.imavs.org/

Figure 5 shows an upper view from the experimental environment and a view from the UAV's camera while flying.



(a) A view of the UAV and the camera frame.



(b) Superior view from the evaluation environment.

Figure 5: The experimental environment.

Multiple flights were performed over an area containing 30 instances of a black-and-white fiducial target, with targets placed in different positions across 3 scenarios and the UAV executing varied trajectories. These scenarios are illustrated in Figure 6, in which each plot shows the targets' geolocations and the vertices of the search area. In addition, 6 different paths were defined for the UAV to follow surrounding the targets' area, totalizing $3 \times 6 = 18$ runs. Figure 7 shows the predefined paths we used in the experiments.

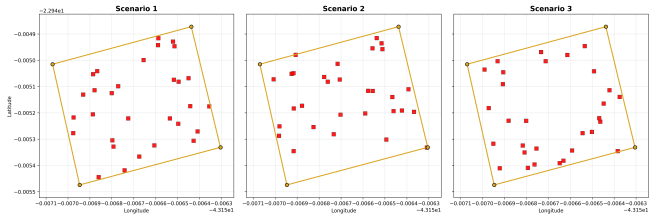
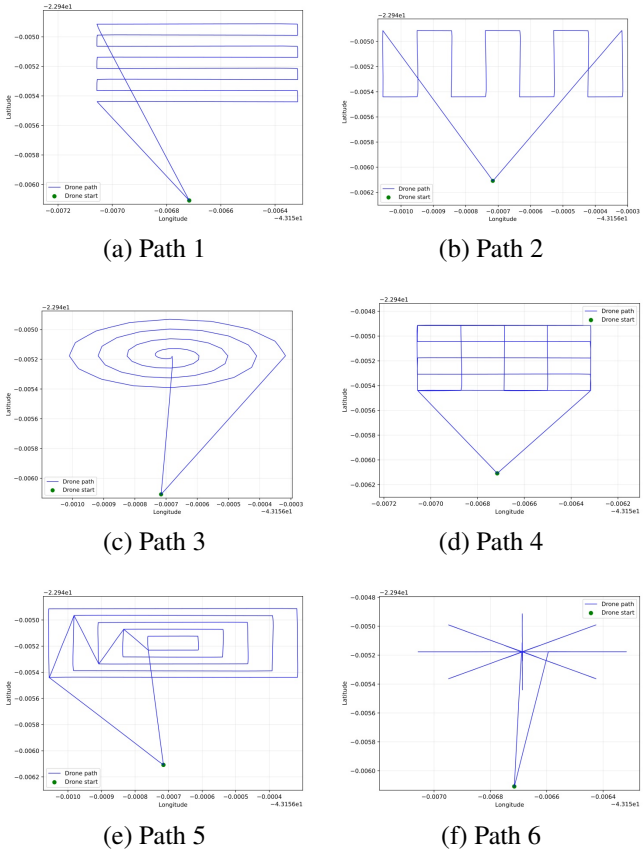


Figure 6: Scenarios of target distributions used in the experiments. The red squares indicate the targets' positions, while the yellow rectangle indicates the boundaries of the targets' area.



(a) Path 1 (b) Path 2

(c) Path 3 (d) Path 4

(e) Path 5 (f) Path 6

Figure 7: Pre-defined UAV paths used in the experiments.

In each run, we saved: the UAV's trajectory; the geocoordinates (lat, long) of each detected object; and the final list of confirmed tracks.

For quantitative evaluation, each confirmed track was matched to its nearest ground-truth target. Regarding these matches, we measured the RMSE between the ground-truth targets and our estimated positions and the detections' recall, i.e., the fraction of real targets successfully identified by the system.

5.2 *Results*

Table 1 presents the quantitative results in terms of localization RMSE (in meters) and the number of correctly identified targets for each flight path and scenario. The results demonstrate the system's capability to perform geolocalization of ground objects using only a monocular camera and the UAV's standard state estimates.

Overall, the system achieved sub-meter localization accuracy in most experiments, with RMSE values ranging from 0.45 m to 0.84 m, demonstrating that the proposed spatial pipeline is capable of consolidating multiple noisy observations. Note that, both the robot's path and the targets' distribution have impacts in the accuracy.

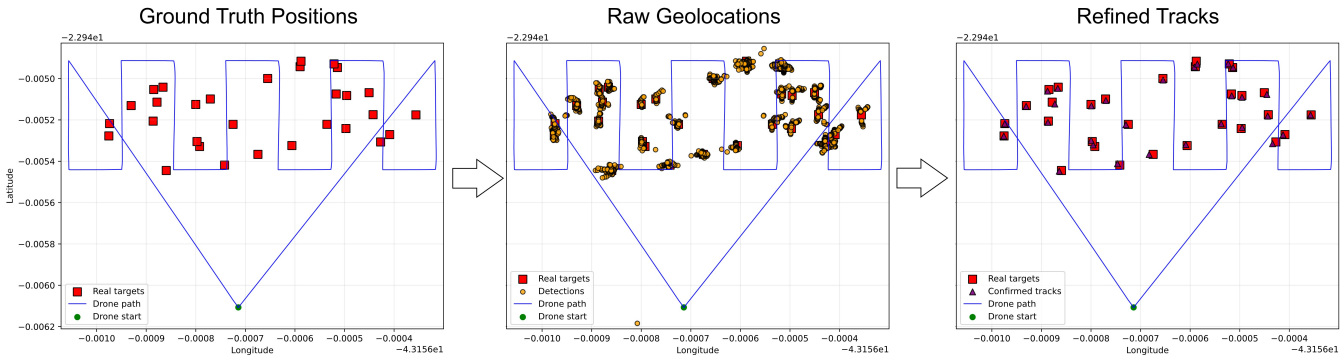


Figure 8: Results from path 2 being executed in scenario 1.

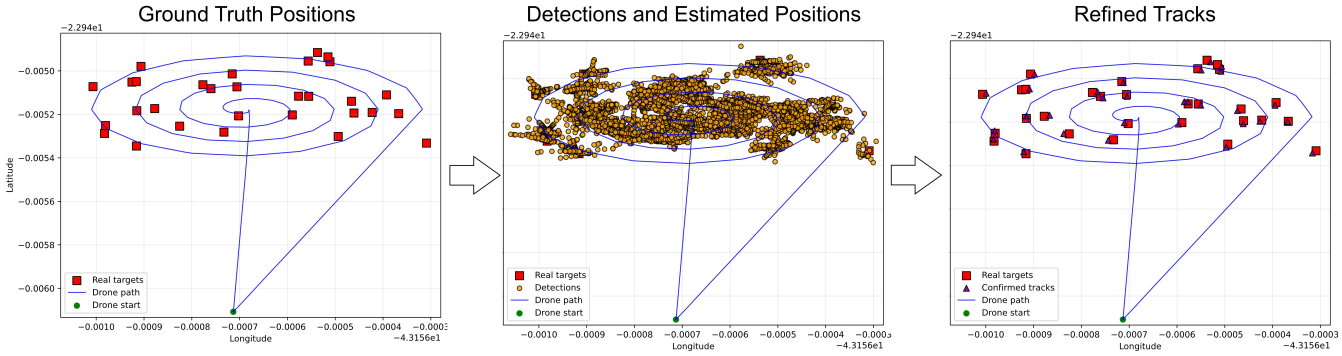


Figure 9: Results from path 3 being executed in scenario 2.

As shown in the Figures 8, 9 and 10, the raw geo-location estimates form highly dispersed point clouds around the true target positions. This dispersion is mainly caused by projection inaccuracies, GPS noise, perspective distortions, false positives from the object detector, and small UAV attitude oscillations. Nevertheless, the proposed tracking and filtering stages progressively reduce this spatial dispersion through repeated Kalman filter updates and trajectory association, resulting in a more reliable representation of the environment.

Table 1: Final geo-localization results (RMSE / Number of Targets per run).

	Scenario (RMSE [m] / Targets)		
Path	**1**	**2**	**3**
1	0.49 / 30	0.52 / 29	0.72 / 30
2	0.59 / 30	0.45 / 29	0.60 / 30
3	0.66 / 30	0.70 / 29	0.78 / 29
4	0.50 / 30	0.47 / 28	0.48 / 29
5	0.84 / 30	0.49 / 28	0.60 / 30
6	0.65 / 29	0.64 / 29	0.67 / 30

The results also show a high target recovery rate, with most runs successfully identifying all 30 targets, as illustrated in Figure 8, which shows the resulting detections and confirmed tracks when Path 2 was executed in Scenario 1. In this experiment, all targets were identified and have their corresponding tracks, with no duplicates. This behavior also happens in Scenario 3, which has one of its runs shown in Figure 10, on which all targets were correctly identified.

Experiments in Scenario 2 consistently presented a lower number of confirmed targets, with several runs detecting only 28 or 29 targets. This behavior is directly related to the spatial arrangement of the targets in this scenario, where multiple targets were positioned extremely close to one another, as shown in Figure 9. Under these conditions, small localization errors become significant enough to cause ambiguity during the association stage. As a consequence, detections originating from neighboring targets may be incorrectly associated with the same trajectory, leading the system to merge two distinct objects into a single confirmed track.

Despite the success in filtering and spatial association, the methodology presents vulnerabilities that affect accuracy. Since the camera is rigidly attached to the drone body without a stabilization system, any perturbation in the UAV attitude directly affects the camera pose, introducing additional errors into the geolocation process. Furthermore, the geometric projection model relies on a flat-ground assumption and is sensitive to inaccuracies in the onboard GPS system.

http://www.imavs.org/

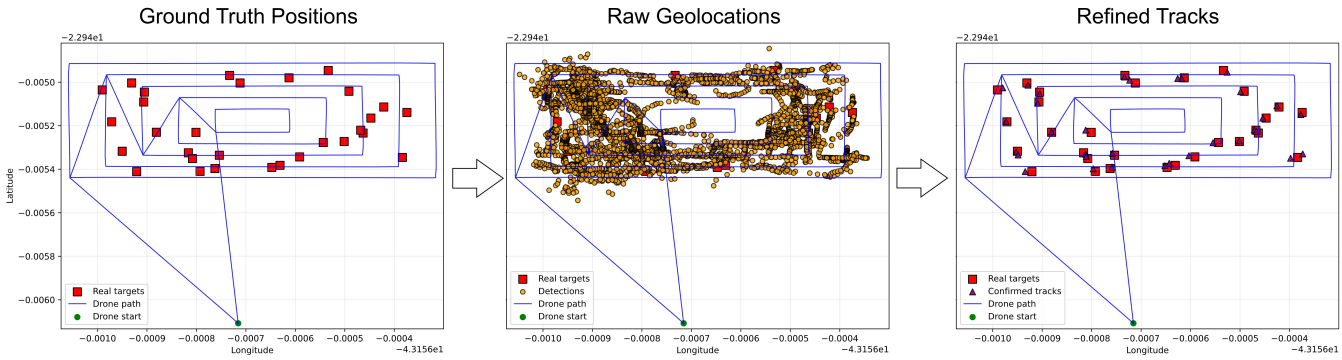


Figure 10: Results from path 5 being executed in scenario 3.

Finally, the system is affected by information latency. Although the pipeline includes a linear interpolation method to mitigate the fact that the flight controller's position and attitude data are published asynchronously, the inherent latency between image acquisition by the camera and the processing of state messages may still generate small temporal misalignments, degrading the estimation performance during high-speed flights or abrupt maneuvers.

6 Conclusion

This paper presented an online pipeline for geo-localizing multiple terrestrial objects from UAV monocular imagery. The proposed approach combines three tightly integrated components: a timestamp interpolation scheme that synchronizes camera frames with asynchronous UAV state estimates, a geometric projection model that maps image detections to ground-plane coordinates using UAV attitude and position, and a spatial refinement stage that employs the Hungarian algorithm and Kalman filtering to associate, filter, and consolidate repeated noisy observations into consistent geolocations. Together, these components enable accurate estimation of ground object positions, eliminating false positives and consolidating repeated observations into consistent geolocations through a spatial refinement pipeline.

Experiments conducted in simulation using Gazebo and ArduPilot SITL across 3 target distribution scenarios and 6 distinct flight paths demonstrated the robustness and accuracy of the approach. The system achieved a mean localization RMSE of 0.603 m while successfully identifying 97.9% of the 30 targets per scenario. These results validate the effectiveness of the spatial refinement pipeline in reducing localization noise and suppressing false positives, even under the constraints of a rigidly mounted camera and a planar ground assumption.

Future work include replacing linear interpolation with higher-order or learning-based state estimation to reduce temporal misalignments during aggressive maneuvers. Additionally, conducting experiments in real outdoor environments will be essential for further validation of the proposed methodology, as well as extending the evaluation to a broader variety of target objects beyond the fiducial markers used in this work.

References

[1] Fangbing Zhang, Tao Yang, Yi Bai, Yajia Ning, Ye Li, Jinghui Fan, and Dongdong Li. Online ground multitarget geolocation based on 3-d map construction using a uav platform. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–17, 2022.

[2] Ziang Ma, Chuanzhi Chen, Jinbao Chen, and Yuhan Jiang. Stc-sort: A dynamic spatio-temporal consistency framework for multi-object tracking in uav videos. *Applied Sciences*, 16(2), 2026.

[3] Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[4] Glenn Jocher and Jing Qiu. Ultralytics yolo11, 2024.

[5] Rahima Khanam and Muhammad Hussain. Yolov11: An overview of the key architectural enhancements, 2024.

[6] Daghash K. Alqahtani, Muhammad Aamir Cheema, and Adel N. Toosi. Benchmarking deep learning models for object detection on edge computing devices. In Walid Gaaloul, Michael Sheng, Qi Yu, and Sami Yangui, editors, *Service-Oriented Computing*, pages 142–150, Singapore, 2025. Springer Nature Singapore.

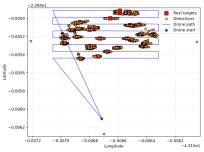
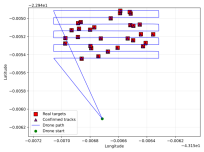
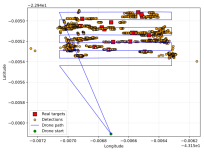
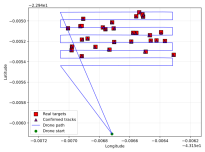
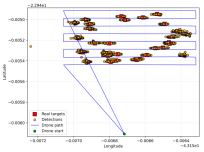
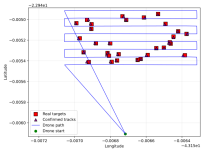
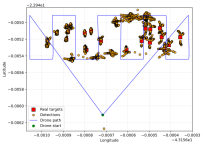
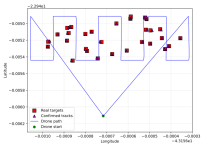
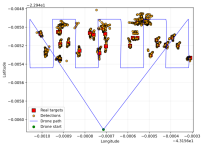
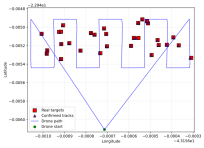
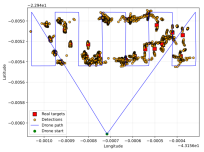
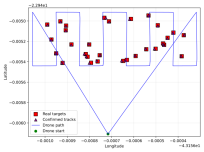
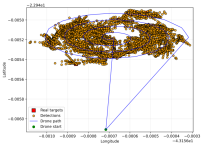
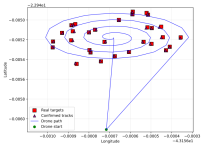
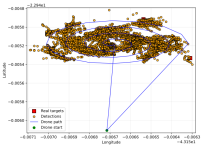
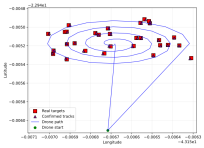
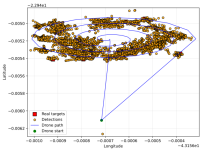
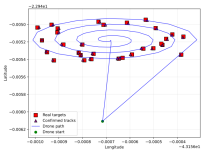
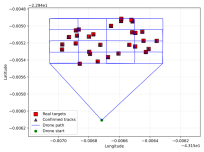
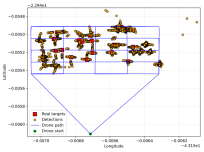
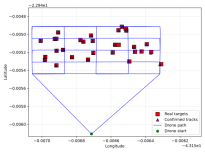
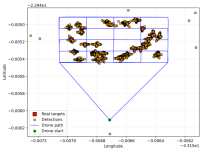
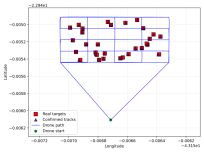
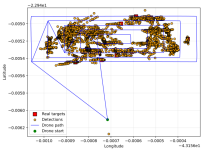
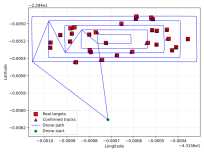
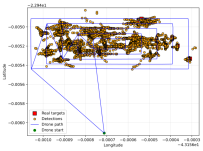
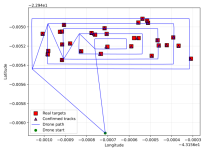
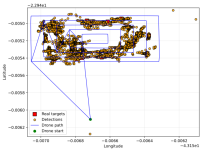
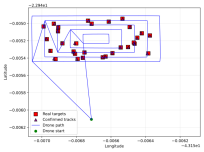
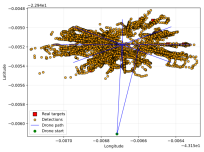
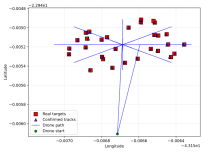
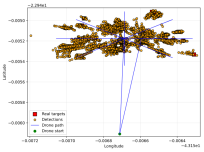
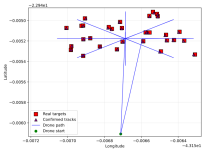
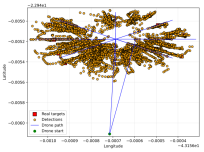
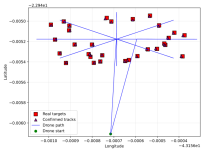
[7] Bruno Eduardo de Oliveira Carvalho, Fabio Luiz Junior, Jauvane Cavalcante de Oliveira, and Paulo Fernando Ferreira Rosa. Lightweight AR System for Precise Object Geolocation in Real-Time Drone Video. In *2025 27th Symposium on Virtual and Augmented Reality (SVR)*, pages 28–38, Los Alamitos, CA, USA, October 2025. IEEE Computer Society.

[8] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, March 2004.

http://www.imavs.org/

Appendix A: Data

Table 2: Comparative analysis of targets detections vs. confirmed tracks across all paths and scenarios.

Path	Scenario 1		Scenario 2		Scenario 3	
	Detections	Conf. Tracks	Detections	Conf. Tracks	Detections	Conf. Tracks
1						
2						
3						
4						
5						
6						

http://www.imavs.org/

Actuator nonlinear fault-tolerant control of a fast-spinning quadrotor with loss of two opposing rotors

T.M.L. De Ponti*, E.J.J. Smeur and B.W.D. Remes
Delft University of Technology, 2629 HS Delft, The Netherlands

http://www.imavs.org/

ABSTRACT

The problem of maintaining control of a quadcopter following the failure of motors is well established within the Fault Tolerant Control (FTC) community. While Incremental Nonlinear Dynamic Inversion (INDI) has been successfully applied to achieve relaxed hover in multirotors under opposing motor failures, its framework typically assumes infinitely fast actuators and neglects state-dependent effects. However, for a quadcopter in a failure scenario where two opposing motors are lost, the vehicle must spin rapidly to stabilize. At high spin rates, the gyroscopic stabilization effect becomes significant, and the assumption of negligible state effects becomes unreasonable. To address this limitation, this paper investigates the use of Actuator Nonlinear Dynamic Inversion (ANDI), a refined control strategy that explicitly incorporates actuator dynamics and state-induced effects into the control allocation. Through theoretical analysis, simulation, and real-world flight tests at the TU Delft indoor test facility, the proposed ANDI controller is directly compared to a baseline INDI controller. The results demonstrate that by compensating for state-induced gyroscopic effects, the ANDI framework is capable of guiding and stabilizing the quadcopter at significantly higher spin rates than INDI. Furthermore, the ANDI controller exhibits better control allocation, evidenced by a lower control effort variance during stationary flight compared to INDI. To the best of the authors' knowledge, this work presents the first successful real-world flight demonstration of an ANDI controller applied to a quadcopter under the failure of two opposing motors.

1 INTRODUCTION

The problem of maintaining control of a quadcopter following the failure of one or more motors is a well-established challenge within the Fault Tolerant Control (FTC) community. [1] first introduced the periodic solution to the so-called reduced-attitude (or relaxed-hover) problem for cases involving one, two, or three failed motors. Subsequently, [2] demonstrated that this solution could be realized in practice using an Linear Quadratic Regulator (LQR). An alternative and highly effective control strategy for the relaxed-hover problem is Incremental Nonlinear Dynamic Inversion (INDI), as proposed by [3] and [4].

INDI has been successfully applied across a broad range of platforms, including multirotors [5], fixed-wing aircraft [6], Vertical Take-Off and Landing (VTOL) vehicles [7], and helicopters [8]. Its primary advantages over traditional controllers include rapid disturbance rejection [5, 6, 9], a reduced need for gain scheduling, and a minimized reliance on accurate model information due to its sensor-based formulation [10]. In [3], these disturbance-rejection capabilities were exploited to achieve relaxed hover in the presence of varying airspeed with a single failed motor. [4] further extended this high-speed INDI FTC framework to accommodate the failure of two motors. More recently, in [11], INDI was applied to control a large quad-plane experiencing a failure of two opposing motors.

In the standard INDI framework, actuator dynamics are not explicitly modeled, and state-dependent effects are typically neglected. This simplification is often justified by assuming that state effects evolve slowly compared to the fast response of the actuators, as discussed in [12]. However, this assumption breaks down when state-induced effects are highly dynamic or when the system utilizes actuators with varying response times.

To address this limitation, the work of [13] reformulated the INDI control law, modifying the control effectiveness matrix used in the allocation process to explicitly account for actuator dynamics [14]. Building on this, the methods presented in [13], [15], and [16] derived the control law by taking an additional time derivative of the system output, seamlessly incorporating actuator dynamics directly into the effectiveness matrix. This refined control strategy, referred to as Actuator Nonlinear Dynamic Inversion (ANDI) in [16], enhances the compensation of actuator lag and improves overall system stability. The framework established in [16] was later extended to more generalized actuator dynamics in [17]. Furthermore, [18] and [19] exploited ANDI to control systems featuring heterogeneous actuator dynamics, demonstrating clear improvements over standard INDI. In contrast, [20] focused on analyzing the benefits of using ANDI to compensate for state effects arising from aerodynamic interactions of a winged platform, concluding that the performance improve-

*Email address: tmldeponti@tudelft.nl

http://www.imavs.org/

ment over INDI was marginal because the aerodynamic state effects were small and evolved slowly.

Despite these advancements, little research has addressed the application of ANDI in scenarios where state-dependent effects are highly significant and evolve as rapidly as the actuator responses themselves. The reduced-attitude control of a quadcopter presents an ideal test case for this problem, as the gyroscopic stabilization effects inherent to a rapidly spinning body cannot be deemed negligible, especially for platforms with specific inertia distributions operating at high spin rates. Crucially, the faster the required spin rate, the closer the frequency of the state-induced effects approaches the bandwidth of the actuator dynamics. Such scenarios are particularly relevant for heavy-lift quadcopters. These platforms typically feature relatively compact, aerodynamic bodies paired with large motors and propellers. In the event of two opposing motors failing, the substantial aerodynamic drag of the propellers requires the drone to spin rapidly to maintain lift. Concurrently, larger motors inherently possess slower dynamic responses, rendering the standard INDI assumption of infinitely fast actuators and negligible state effects unreasonable.

To address this gap in the literature, the main contributions of this paper are twofold: (1) to the best of the authors' knowledge, this work presents the first successful real-world flight demonstration of an ANDI controller applied to a quadcopter experiencing a complete failure of two opposing motors; (2) a comprehensive comparative analysis between the proposed ANDI framework and a baseline INDI controller, demonstrating the operational limits and control allocation benefits of explicitly accounting for gyroscopic state effects.

2 SYSTEM MODEL

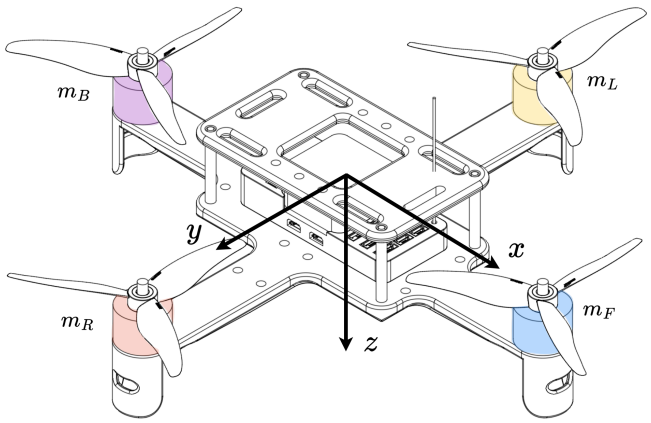


Figure 1: The quadcopter used in the experiments with body axis definition.

The drone utilized in the experiments, along with the body axis definition, is shown in Figure 1. The drone is equipped with four identical motors, denoted as m_F, m_R, m_B, and m_L for the front, right, back, and left motors, respectively. It is assumed that the motors exhibit first-order dynamics with corner freqeuncy ε. Furthermore, the thrust generated by each motor is modeled as being proportional to the square of its rotational speed, such that:

$$T_i = k_T \omega_i^2, \tag{1}$$

where T_i is the thrust generated by the i-th motor, k_T is the thrust coefficient, and ω_i is the rotational speed. Consequently, the control force $\boldsymbol{F}_c$ exerted in the body frame is given by:

$$\begin{bmatrix} F_{c_x} \\ F_{c_y} \\ F_{c_z} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -k_T & -k_T & -k_T & -k_T \end{bmatrix} \begin{bmatrix} \omega_F^2 \\ \omega_R^2 \\ \omega_B^2 \\ \omega_L^2 \end{bmatrix}. \tag{2}$$

All motors are located at the same distance l from the center of gravity. Thus, the resulting control moment $\boldsymbol{M}_c$ exerted in the body frame is modeled as:

$$\begin{bmatrix} M_{c_x} \\ M_{c_y} \\ M_{c_z} \end{bmatrix} = \begin{bmatrix} 0 & -lk_T & 0 & lk_T \\ lk_T & 0 & -lk_T & 0 \\ -\tau & \tau & -\tau & \tau \end{bmatrix} \begin{bmatrix} \omega_F^2 \\ \omega_R^2 \\ \omega_B^2 \\ \omega_L^2 \end{bmatrix}. \tag{3}$$

Here, the reaction torque produced by the motors is also modeled as being proportional to the square of the rotational speed, with a proportionality coefficient τ. The inertia tensor of the platform is assumed diagonal with principal moments of inertia I_{xx}, I_{yy}, and I_{zz}. Finally Table 1 reports the parameters of the model of the platform.

Parameter	Vaule	Unit
m	9.38e-1	kg
I_{xx}	2.70e-3	$\mathrm{kg\,m^2}$
I_{yy}	3.60e-3	$\mathrm{kg\,m^2}$
I_{zz}	4.50e-3	$\mathrm{kg\,m^2}$
k_T	7.564e-7	$\mathrm{N\,s^2\,rad^{-2}}$
τ	7.657e-9	$\mathrm{N\,m\,s^2\,rad^{-2}}$
ε	29.0	$\mathrm{rad\,s^{-1}}$
l	1.30e-1	m

Table 1: Parameters of the quadcopter.

3 CONTROL LAWS

Both the ANDI and INDI control laws, as for example shown in [11] and [18],can be derived for a general nonlinear system by assuming first-order actuator dynamics and infinitely fast actuators, respectively. Consider a system with output $\boldsymbol{y} \in \mathbb{R}^m$, state vector $\boldsymbol{x} \in \mathbb{R}^k$, and actuator state vector $\boldsymbol{u} \in \mathbb{R}^n$, governed by:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), \quad \boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}). \tag{4}$$

http://www.imavs.org/

The system has a relative degree r with respect to $\boldsymbol{u}$ (i.e., the output $\boldsymbol{y}$ must be differentiated r times before the input $\boldsymbol{u}$ appears explicitly [16]), such that:

$$\boldsymbol{y}^{(r)} = \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{u}). \tag{5}$$

Taking an additional time derivative yields:

$$\boldsymbol{y}^{(r+1)} = \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{x}}\,\dot{\boldsymbol{x}} + \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}}\,\dot{\boldsymbol{u}} = \boldsymbol{F}_x\,\dot{\boldsymbol{x}} + \boldsymbol{F}_u\,\dot{\boldsymbol{u}}. \tag{6}$$

For notational simplicity, we assume $r = 1$. Next, consider stable actuator dynamics of degree d, described by the transfer function:

$$A = \frac{U}{U_c} = \frac{\prod_{i=1}^{d}(-\varepsilon_i)}{\prod_{i=1}^{d}(s - \varepsilon_i)}, \tag{7}$$

where ε_i denotes the i-th pole with $\Re\{\varepsilon_i\} < 0$, and capital letters indicate variables expressed in the Laplace domain. Here, U_c denotes the commanded actuator input and U the realized actuator state after actuator dynamics. For the multiple input multiple output (MIMO) case, $\boldsymbol{A} \in \mathbb{R}\,(s)^{n\times n}$ is a diagonal matrix of transfer functions. The incremental commanded control input can then be defined in terms of the actuator dynamics as:

$$\boldsymbol{\Delta U} = \boldsymbol{U_c} - \boldsymbol{U} = \big(\boldsymbol{I} - \boldsymbol{A}\big)\,\boldsymbol{U_c}. \tag{8}$$

From Eq. (8), it follows that:

$$\mathcal{L}\{\dot{\boldsymbol{u}}\} = s\boldsymbol{U} = s\boldsymbol{A}\boldsymbol{U_c} = s\boldsymbol{A}\,[\boldsymbol{I} - \boldsymbol{A}]^{-1}\,\boldsymbol{\Delta U} = \boldsymbol{W}\,\boldsymbol{\Delta U}. \tag{9}$$

Let the pseudo-control vector $\boldsymbol{\nu} \in \mathbb{R}^m$ (denoted as $\boldsymbol{N}$ in the Laplace domain) represent the desired system output $\boldsymbol{y}_d$. Additionally, we define the transfer function matrix $\boldsymbol{V} \in \mathbb{R}\,(s)^{m\times m}$, which maps the tracking error $\boldsymbol{e} = \boldsymbol{\nu} - \boldsymbol{y}$ to $\dot{\boldsymbol{y}}$:

$$\mathcal{L}\{\dot{\boldsymbol{y}}\} = s\boldsymbol{Y} = \boldsymbol{V}\big(\boldsymbol{N} - \boldsymbol{Y}\big). \tag{10}$$

To maximize performance without overdriving the actuators, a natural design choice is to match the desired closed-loop output dynamics along each axis directly to the bandwidth of the slowest influencing actuator. Substituting $Y = AN$ into Eq. (10) for each individual control axis and rearranging gives:

$$sAN = V\big(1 - A\big)N \Longleftrightarrow V = sA(1 - A)^{-1} = W. \tag{11}$$

Assuming $\boldsymbol{F}_x$ and $\boldsymbol{F}_u$ are approximately constant over one control step, we can substitute Eqs. (9) to (11) into Eq. (6) and rearrange to obtain the general closed-loop expression:

$$\begin{aligned} \boldsymbol{V}\big(\boldsymbol{N} - \boldsymbol{Y}\big) &= \boldsymbol{F}_u\boldsymbol{W}\boldsymbol{\Delta U} + \boldsymbol{F}_x s\boldsymbol{X}, \\ \boldsymbol{\Delta U} &= \big(\boldsymbol{F}_u\boldsymbol{W}\big)^{+}\Big(\boldsymbol{V}\big(\boldsymbol{N} - \boldsymbol{Y}\big) - \boldsymbol{F}_x s\boldsymbol{X}\Big). \end{aligned} \tag{12}$$

3.1 INDI

The INDI approach assumes infinitely fast actuators. Equation (7) implies that as the actuator dynamics become infinitely fast (i.e., $\Re\{\varepsilon_i\} \to -\infty$), $A \to 1$, which consequently means $W \to \infty$ and $V \to \infty$. Under this assumption, Eq. (12) reduces to the INDI control law:

$$\begin{aligned} \boldsymbol{\Delta U} &= \big(\boldsymbol{F}_u\big)^{+}\big(\boldsymbol{N} - \boldsymbol{Y}\big), \\ \boldsymbol{\Delta u} &= \big(\boldsymbol{F}_u\big)^{+}\big(\boldsymbol{\nu} - \boldsymbol{y}\big). \end{aligned} \tag{13}$$

In this control law, $\boldsymbol{F}_u$ is commonly referred to as the control effectiveness matrix.

3.2 ANDI

In contrast, ANDI explicitly accounts for first-order actuator dynamics:

$$A = \frac{\varepsilon}{s + \varepsilon}. \tag{14}$$

It follows that $\boldsymbol{W}$ is a diagonal matrix containing the corner frequency of each actuator:

$$\boldsymbol{W} = \begin{bmatrix} \varepsilon_{u_1} & & \\ & \ddots & \\ & & \varepsilon_{u_n} \end{bmatrix}, \tag{15}$$

while $\boldsymbol{V}$ is a diagonal matrix containing the corner frequency of the slowest actuator associated with each controlled axis:

$$\boldsymbol{V} = \begin{bmatrix} \varepsilon_{\nu_1} & & \\ & \ddots & \\ & & \varepsilon_{\nu_m} \end{bmatrix}. \tag{16}$$

For a fully actuated system with identical actuators, it is clear that $\boldsymbol{W} = \boldsymbol{V}$, which simplifies the control law to:

$$\begin{aligned} \boldsymbol{\Delta U} &= \big(\boldsymbol{F}_u\big)^{+}\,(\boldsymbol{N} - \boldsymbol{Y}) - \big(\boldsymbol{F}_u\boldsymbol{W}\big)^{+}\,(\boldsymbol{F}_x s\boldsymbol{X})\,, \\ \boldsymbol{\Delta u} &= \big(\boldsymbol{F}_u\big)^{+}\,(\boldsymbol{\nu} - \boldsymbol{y}) - \big(\boldsymbol{F}_u\boldsymbol{W}\big)^{+}\,(\boldsymbol{F}_x\dot{\boldsymbol{x}})\,. \end{aligned} \tag{17}$$

Equation (17) highlights a fundamental operational difference between ANDI and INDI: ANDI explicitly compensates for the state-driven term $\boldsymbol{F}_x s\boldsymbol{X}$ in its control allocation.

4 STATE EFFECT

To illustrate the effect of neglecting the state-driven term, consider the problem of controlling the roll angular acceleration of a spinning gyroscope with the same physical parameters of the test platform as reported in Section 2. A constant yaw moment M_z is applied to the gyroscope, resulting in a continuously increasing yaw rate. The controller aims to track a desired roll angular acceleration $\dot{p}_d$ by manipulating the moment exerted about the body x-axis, M_x. This moment is generated by a first-order actuator with corner frequency ε and is proportional to its state u, such that

$$M_x = k_x u. \tag{18}$$

Aerodynamic effects and other disturbances are neglected, so that $M_y = 0$. From Euler's equations for the rotational motion of a rigid body, expressed in the body frame, it follows that

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}\frac{k_x u}{I_{xx}} - \frac{I_{zz}-I_{yy}}{I_{xx}}qr\\ -\frac{I_{xx}-I_{zz}}{I_{yy}}pr\\ M_z - \frac{I_{yy}-I_{xx}}{I_{zz}}pq\end{bmatrix} = \begin{bmatrix}\frac{k_x u}{I_{xx}} + C_x qr\\ C_y pr\\ M_z + C_z pq\end{bmatrix}. \quad (19)$$

Taking one extra derivative in time leads to:

$$\begin{bmatrix}\ddot{p}\\ \ddot{q}\\ \ddot{r}\end{bmatrix} = \underbrace{\begin{bmatrix}\frac{k_x}{I_{xx}}\\ 0\\ 0\end{bmatrix}}_{F_u}\dot{u} + \underbrace{\begin{bmatrix}0 & C_x r & C_x q\\ C_y r & 0 & C_y p\\ C_z q & C_z p & 0\end{bmatrix}}_{F_x}\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix}. \quad (20)$$

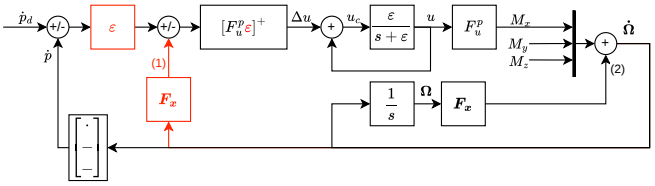


Figure 2: Structure for the gyro controller. Red signals and blocks are only present in ANDI.

The control structure of an ANDI controller (red signals included) is shown in Figure 2. It uses the control law of Eq. (17), designed to track a desired roll angular acceleration using the roll effectiveness F_u^p, i.e., the entry of $\boldsymbol{F_u}$ corresponding to the roll axis. In particular, by rearranging the diagram, it can be shown that the state compensation signal (1) cancels the state effect on the controlled signal (2). In other words, because the state compensation is embedded directly in the control law, ANDI tracks $\dot{p}_d$ with the same dynamics regardless of the gyroscope spin rate. More precisely, as discussed in Section 3, these dynamics are equal to the actuator dynamics.

Figure 2 also shows the control structure of an INDI controller (red signals excluded), which uses the control law of Eq. (13), designed to perform the same task. It is clear that, unless $\boldsymbol{F_x}$ or $\boldsymbol{\Omega}$ is zero, the closed-loop dynamics differ from the actuator dynamics. The controller allocates the actuator input while completely neglecting the effect of the state on the controlled signal. As the yaw rate r increases, the gyroscopic coupling becomes progressively stronger, and neglecting this effect is therefore expected to degrade performance.

This conclusion is confirmed by the simulation shown in Figure 3, in which the gyroscope spins increasingly faster about the yaw axis. ANDI is able to track the desired sinusoidal $\dot{p}$ accurately, with dynamics matching those of the actuator $\dot{p}_{dyn}$, whereas INDI exhibits increasing attenuation and lead as r grows. The INDI behavior can also be confirmed analytically by linearizing the nonlinear plant around

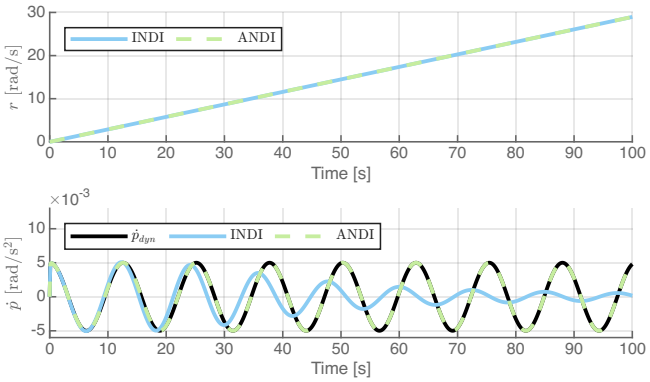


Figure 3: Simulation of a spinning gyroscope.

a trim condition $z = [p_0, q_0, r_0]^T$, where r_0 is constant and non-zero and the remaining angular rates are zero. This allows the linearized dynamics to be expressed in state-space form as:

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \underbrace{\begin{bmatrix}0 & C_x r_0 & 0\\ C_y r_0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}}_{A}\begin{bmatrix}p\\ q\\ r\end{bmatrix} + \underbrace{\begin{bmatrix}\frac{k_x}{I_{xx}}\\ 0\\ 0\end{bmatrix}}_{B}u. \quad (21)$$

The resulting transfer function for the tracking dynamics of the INDI controller is given by:

$$\frac{\dot{p}}{\dot{p}_d}(s)_I = \frac{\varepsilon\, s}{s^2 + \varepsilon\, s - C_x\, C_y\, {r_0}^2}. \quad (22)$$

Equation (22) shows how the INDI controller behaves as a second-order system with a zero at the origin and a pair of complex conjugate poles. As r_0 increases, the poles move further to the right in the complex plane, resulting in increased attenuation and phase lead at frequencies below the actuator bandwidth. Such system reduces to the actuator dynamics only when $r_0 = 0$, which is the condition under which the state effect is negligible and the INDI assumption holds.

A Bode plot for the transfer function in Eq. (22) is shown in Figure 4. It confirms that as r_0 becomes larger, the controller increasingly attenuates the signal and introduces phase lead at frequencies below the actuator's bandwidth.

Consequently, for a rapidly spinning body, such as a gyroscope or a quadcopter experiencing a failure of two opposing motors, the INDI approach is expected to suffer from progressively degraded performance as the spin rate increases. In contrast, ANDI will properly compensate for these state-dependent effects and maintain the expected actuator dynamics.

5 REDUCED ATTITUDE

When a quadcopter loses two opposing motors, it becomes underactuated and can no longer independently control all its attitude degrees of freedom. Typically, yaw control

http://www.imavs.org/

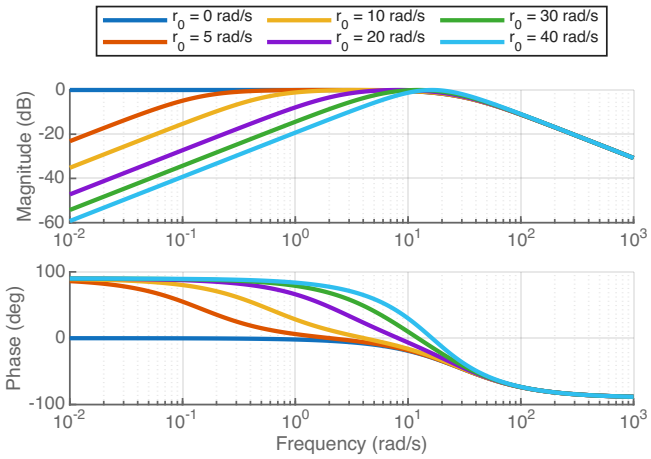


Figure 4: Bode plot of the INDI dynamics at different yaw rates.

is sacrificed, causing the vehicle to spin continuously about its vertical axis. This necessitates a control framework that can still achieve stabilization independently of the yaw angle, a method commonly known as the reduced attitude control problem.

The key idea is to regulate the principal axis of rotation, expressed in the body frame as $\boldsymbol{\mu}^B \in \mathbb{R}^3$, such that it tracks a desired inertial direction $\boldsymbol{n}^I \in \mathbb{R}^3$. To apply ANDI or INDI to this problem, we differentiate the relevant kinematics until the actuator states appear explicitly, and then differentiate once again with respect to time, as shown in Eq. (6).

Let $\boldsymbol{n}^I$ and $\boldsymbol{n}^B$ denote the same geometric vector expressed in the inertial and rotating body frames respectively such that:

$$\boldsymbol{n}^B = \boldsymbol{L}_{BI}\, \boldsymbol{n}^I, \tag{23}$$

where $\boldsymbol{L}_{BI}$ is the rotation matrix from the inertial frame to the body frame. The time derivative of $\boldsymbol{n}^B$ observed in the inertial frame is given by: The time derivatives of a generic vector $\boldsymbol{Q}$ observed in these frames are related by the transport theorem:

$$\left(\left.\frac{d\boldsymbol{Q}}{dt}\right|_I\right)^B = \left(\left.\frac{d\boldsymbol{Q}}{dt}\right|_B\right)^B + \boldsymbol{\omega}^B \times \boldsymbol{Q}^B, \tag{24}$$

where the superscript indicates the frame in which the quantity is expressed, and the subscript indicates the frame with respect to which the derivative is taken. Higher-order derivatives are obtained through repeated application of Eq. (24). Compactly, the m-th derivative is:

$$\left(\left.\frac{d^m\boldsymbol{Q}}{dt^m}\right|_I\right)^B = \left[\frac{d(*)}{dt} + \boldsymbol{\omega}^B \times (*)\right]^{(m)} \boldsymbol{Q}^B, \tag{25}$$

where $[\,\cdot\,]^{(m)}$ denotes m successive applications of the operator in brackets.

Using this result, one can derive expressions for the time derivatives of $\boldsymbol{n}^B$ as observed in the body frame. For the first derivative:

$$(\dot{\boldsymbol{n}}|_B)^B = -\boldsymbol{\omega}^B \times \boldsymbol{n}^B + (\dot{\boldsymbol{n}}|_I)^B, \tag{26}$$

and for the second derivative:

$$\begin{aligned}(\ddot{\boldsymbol{n}}|_B)^B = &-(\dot{\boldsymbol{\omega}}|_B)^B \times \boldsymbol{n}^B + \boldsymbol{\omega}^B \times \left(\boldsymbol{\omega}^B \times \boldsymbol{n}^B\right) \\ &- 2\,\boldsymbol{\omega}^B \times (\dot{\boldsymbol{n}}|_I)^B + (\ddot{\boldsymbol{n}}|_I)^B.\end{aligned} \tag{27}$$

The term $-(\dot{\boldsymbol{\omega}}|_B)^B \times \boldsymbol{n}^B$ can be written more conveniently in matrix form by introducing the skew-symmetric cross-product matrix associated with $\boldsymbol{n}^B$. Defining

$$\boldsymbol{H}_B = [\boldsymbol{n}^B]_\times = \begin{bmatrix} 0 & -n_z^B & n_y^B \\ n_z^B & 0 & -n_x^B \\ -n_y^B & n_x^B & 0 \end{bmatrix}, \tag{28}$$

and using the property $\boldsymbol{a} \times \boldsymbol{b} = [\boldsymbol{a}]_\times \boldsymbol{b}$, we obtain:

$$-(\dot{\boldsymbol{\omega}}|_B)^B \times \boldsymbol{n}^B = \boldsymbol{n}^B \times (\dot{\boldsymbol{\omega}}|_B)^B = \boldsymbol{H}_B\, \dot{\boldsymbol{\omega}}^B. \tag{29}$$

From Euler's equations for the rotational motion of a rigid body about a fixed point, expressed in the body frame, we have:

$$\boldsymbol{I}\, \dot{\boldsymbol{\omega}}^B = \boldsymbol{M}(\boldsymbol{u}, \boldsymbol{x}) - \boldsymbol{\omega}^B \times \left(\boldsymbol{I}\boldsymbol{\omega}^B\right), \tag{30}$$

which yields:

$$\dot{\boldsymbol{\omega}}^B = \boldsymbol{I}^{-1}\boldsymbol{M}(\boldsymbol{u}, \boldsymbol{x}) - \boldsymbol{I}^{-1}\left(\boldsymbol{\omega}^B \times \left(\boldsymbol{I}\boldsymbol{\omega}^B\right)\right), \tag{31}$$

where $\boldsymbol{I}$ is the inertia tensor and $\boldsymbol{M}$ is the applied moment. Finally, the full kinematics of the reduced attitude can be expressed by substituting Eq. (31) into Eq. (27).

Both the INDI and ANDI control laws (Eq. (13) and Eq. (17), respectively) require an expression for the control effectiveness matrix $\boldsymbol{F}_{u_{n^B}}$ to regulate $(\ddot{\boldsymbol{n}}|_B)^B$. Specifically, differentiating the term $\boldsymbol{H}_B\dot{\boldsymbol{\omega}}^B$ with respect to the control input yields:

$$\boldsymbol{F}_{u_{n^B}} = \frac{\partial\,(\ddot{\boldsymbol{n}}|_B)^B}{\partial \boldsymbol{u}} = \boldsymbol{H}_B\, \frac{\partial \dot{\boldsymbol{\omega}}^B}{\partial \boldsymbol{u}} = \boldsymbol{H}_B\, \boldsymbol{I}^{-1}\, \frac{\partial \boldsymbol{M}(\boldsymbol{u}, \boldsymbol{x})}{\partial \boldsymbol{u}}. \tag{32}$$

To apply ANDI, the state compensation term is also required. This is retrieved via:

$$\boldsymbol{F}_{x_{n^B}} = \frac{\partial\,(\ddot{\boldsymbol{n}}|_B)^B}{\partial \boldsymbol{x}_{\boldsymbol{n^B}}} \tag{33}$$

where

$$\boldsymbol{x}_{\boldsymbol{n^B}} = \left[\boldsymbol{\omega^B}, \boldsymbol{L_{BI}}, \boldsymbol{n}^I, (\dot{\boldsymbol{n}}|_I)^I, (\ddot{\boldsymbol{n}}|_I)^I\right]^T. \tag{34}$$

It then follows that:

$$\begin{aligned}\boldsymbol{F}_{x_{n^B}}\dot{\boldsymbol{x}}_{\boldsymbol{n^B}} = &\,\boldsymbol{\omega}^B \times \left(\dot{\boldsymbol{\omega}}^B \times \boldsymbol{n}^B\right) + 2\dot{\boldsymbol{\omega}}^B \times \left(\boldsymbol{\omega}^B \times \boldsymbol{n}^B\right) \\ &- \boldsymbol{\omega}^B \times \left(\boldsymbol{\omega}^B \times \left(\boldsymbol{\omega}^B \times \boldsymbol{n}^B\right)\right) - 3\dot{\boldsymbol{\omega}}^B \times (\dot{\boldsymbol{n}}|_I)^B \\ &+ 3\boldsymbol{\omega}^B \times \left(\boldsymbol{\omega}^B \times (\dot{\boldsymbol{n}}|_I)^B\right) - 3\boldsymbol{\omega}^B \times (\ddot{\boldsymbol{n}}|_I)^B + (\dddot{\boldsymbol{n}}|_I)^B \\ &- \boldsymbol{I}^{-1}\left[\dot{\boldsymbol{\omega}}^B \times \left(\boldsymbol{I}\boldsymbol{\omega}^B\right) - \boldsymbol{\omega}^B \times \left(\boldsymbol{I}\dot{\boldsymbol{\omega}}^B\right)\right].\end{aligned} \tag{35}$$

http://www.imavs.org/

http://www.imavs.org/

Finally, it is important to note that an INDI or ANDI reduced attitude controller is not strictly limited to fault-case scenarios; it can also be used under nominal conditions. A standard nominal controller allocates for the body z-axis acceleration a_z alongside the angular accelerations $\dot{p}$, $\dot{q}$, and $\dot{r}$. By simply replacing the targets for $\dot{p}$ and $\dot{q}$ with the x and y components of $(\ddot{\boldsymbol{n}}|_B)^B$, the modified controller not only retains full nominal flight capabilities but also inherently supports the stabilization of spinning bodies. However, employing this relaxed attitude structure requires a different Error Controller (EC) than that of a nominal controller, which will be discussed in the next section.

6 Controller Design

The controller, whether based on INDI or ANDI, adopts a cascaded structure consisting of an inner stabilization loop and an outer guidance loop. The objective of the stabilization controller is to realize a desired linear acceleration vector. To achieve this, its first task is to align the principal axis of rotation $\boldsymbol{\mu}^B$ with the desired inertial unit vector $\boldsymbol{n}^I$, which represents the direction of the desired specific force expressed in the inertial frame. This principal axis arises from the relaxed-hover equilibrium solution and, in the case of two opposing motor failures, is aligned with the body z-axis, i.e., $\boldsymbol{\mu}^B = [0, 0, -1]^T$ [1]. The controller must then ensure that the magnitude of the generated acceleration is also correct by modulating the total thrust. Finally, when the vehicle is not operating under fault conditions, a desired yaw acceleration can also be tracked. Hence, the subset of controlled variables, for which desired inputs are provided in the pseudo-control vector for both control laws, is:

$$\boldsymbol{x} = \left[a_z, (\ddot{n}|_B)^B_x, (\ddot{n}|_B)^B_y, \dot{r}\right]^T. \tag{36}$$

The innermost control loop in which the two control laws are applied to regulate $\boldsymbol{x}$ is shown in Figure 5. The signals shown in red are present only in the ANDI implementation. The control increment $\boldsymbol{\Delta u}$ is computed through a Weighted Least Squares (WLS) allocator [21, 22]. This routine is commonly used to handle actuator saturation while also establishing a hierarchy among control objectives through the pseudo-control weighting matrix $\boldsymbol{W_v}$. By setting specific diagonal entries of $\boldsymbol{W_v}$ to zero, the corresponding control axes are effectively relaxed in the WLS cost function. This makes it possible to discard axes that cannot be controlled under fault conditions. For example, yaw is always sacrificed, while either $(\ddot{n}|_B)^B_x$ or $(\ddot{n}|_B)^B_y$ is relaxed depending on whether the failed motors correspond to the roll or pitch axis.

As for the computation of $\boldsymbol{\nu}$, a linear error controller

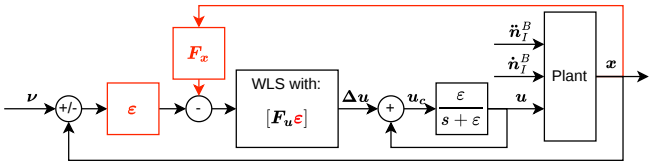


Figure 5: Allocation diagram. Red signals and variables are additional in ANDI compared to INDI.

based on Eq. (23) and Eq. (26) is used, such that:

$$\boldsymbol{\nu} = \begin{bmatrix} a_{z_d} \\ (\ddot{n}|_B)^B_{x_d} \\ (\ddot{n}|_B)^B_{y_d} \\ \dot{r}_d \end{bmatrix} = \begin{bmatrix} a_{z_d} \\ k_1^s k_2^s \left(\mu_x^B - n_x^B\right) - k_2^s \left(\dot{n}|_B\right)^B_x \\ k_1^s k_2^s \left(\mu_y^B - n_y^B\right) - k_2^s \left(\dot{n}|_B\right)^B_y \\ k_1^s k_2^s \left(\psi_d - \psi\right) - k_2^s r \end{bmatrix}. \tag{37}$$

The desired acceleration magnitude a_{z_d}, together with its direction, is computed by the outer-loop guidance controller shown in Fig. 6.

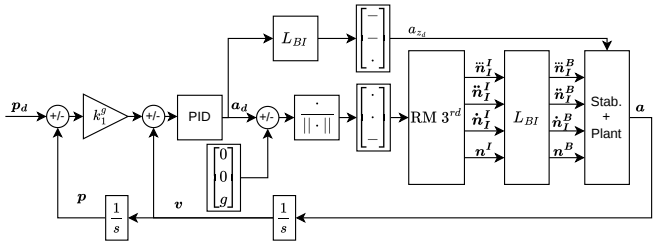


Figure 6: Guidance controller.

The proposed method employs a Reference Model (RM) that maps the desired n^I into smooth, time-synchronized reference trajectories for the desired principal axis and its derivatives. This avoids relative delays between the signals, which is critical for state estimation within the stabilization controller. Moreover, the guidance structure consists of a velocity PID loop nested inside an outer proportional position loop. Finally, Table 2 reports the gains used in the presented control loops.

k_1^s	k_2^s	k_1^g	k_P^g	k_I^g	k_D^g
4.2	10.0	0.6	1.8	0.4	0.2

Table 2: Parameters used in the controllers.

7 Flight Tests

To evaluate the ANDI and INDI controllers, and to analyze their differences in order to draw meaningful conclusions, flight tests were conducted at the indoor test facility of TU Delft, shown in Fig. 7. Attitude and position are estimated using an external motion-capture system running at approximately 80 Hz. All data files and processing scripts are accessible at [23].



Figure 7: Test platform flying in the Cyberzoo, the indoor test facility of TU Delft.

The failure scenario considered in this work is a complete loss of active control authority of the front and back motors. The left and right motors are then responsible for stabilizing and guiding the platform under fault conditions, while the drone spins about its body z-axis. In contrast, the front and back motors are only passively modulated to produce a faster or slower spin and are not used for stabilization; indeed, they are always commanded identically.

The first test is designed to identify the yaw rate at which each controller loses the ability to stabilize the drone. To this end, the drone is commanded to hold a fixed position in space while the front and back motors are slowly commanded a continuously decreasing setpoint. Figure 8 and Figure 9 show the motor commands and the resulting yaw rate for the INDI and ANDI controllers, respectively.

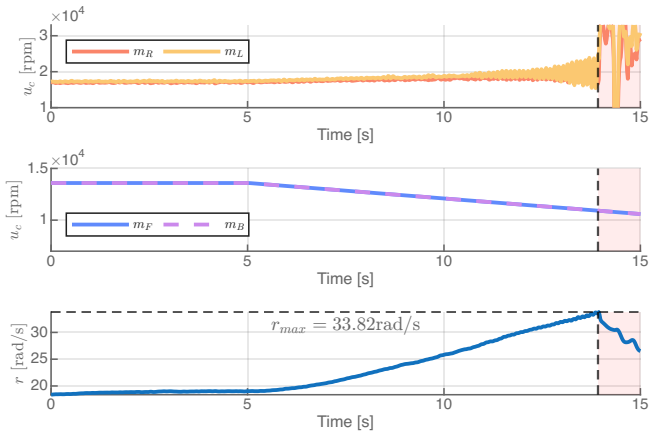


Figure 8: Motor commands and yaw rate in the loss of control test of the INDI controller. The red area indicates that loss of control has occurred.

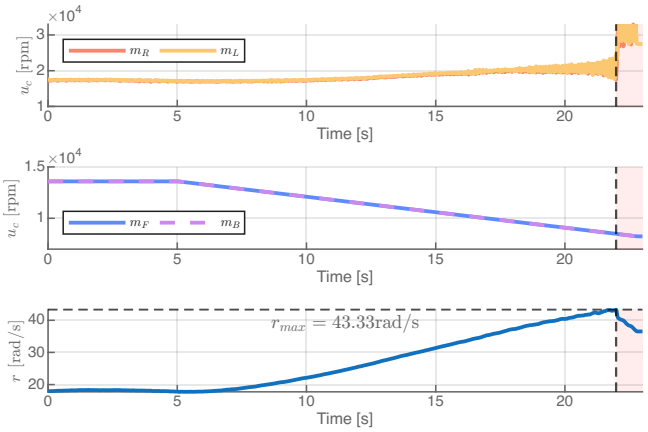


Figure 9: Motor commands and yaw rate in the loss of control test of the ANDI controller. The red area indicates that loss of control has occurred.

For both controllers, the steadily decreasing command to the pitch motors produces a progressively increasing yaw rate. Notably, the INDI controller loses stability at $33.82\,\mathrm{rad\,s^{-1}}$, while the ANDI controller remains stable up to a considerably higher rate of $43.33\,\mathrm{rad\,s^{-1}}$. As the drone spins faster, the gyroscopic coupling induced by the state grows larger, and failing to account for it in the INDI control law leads to a progressive degradation in allocation accuracy. The ANDI controller, by explicitly compensating for this term, is able to maintain stable flight at higher spin rates. Nevertheless, ANDI eventually also experiences a loss of control, which is to be expected. As discussed in Section 4, ANDI does not alter the actuator dynamics; rather, it ensures that the innermost allocation loop follows those dynamics. It is worth noting that the spin rate of $43.33\,\mathrm{rad\,s^{-1}}$ already exceeds the $29\,\mathrm{rad\,s^{-1}}$ corner frequency of the actuator dynamics, at which point the actuator transfer function exhibits an attenuation of $-4.9\,\mathrm{dB}$ and a phase delay of $55.4\,^\circ$. As the spin rate increases further beyond this threshold, the attenuation and phase delay grow, and the ANDI controller will inevitably also fail.

Another conclusion drawn from the test is that both controllers are expected to successfully stabilize the drone at lower spin rates. It is therefore interesting to analyze whether there are noticeable differences between the two controllers in such conditions. To investigate this, the performance of both controllers is compared in a task involving flying a square trajectory at a yaw rate of approximately $20\,\mathrm{rad\,s^{-1}}$, which is achieved by commanding a constant setpoint to the pitch motors. Figure 10 shows the position tracking and the resulting yaw rate for both the INDI and ANDI controllers.

As expected, Fig. 10 demonstrates that both controllers are capable of accurately tracking the square trajectory at a spin rate of roughly $20\,\mathrm{rad\,s^{-1}}$. Furthermore, Fig. 11 and Fig. 12 reveal that this performance is enabled by the accu-

http://www.imavs.org/

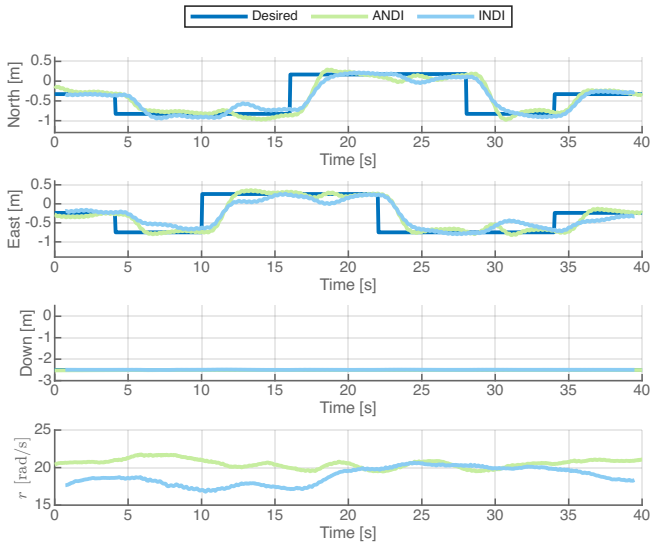


Figure 10: Position tracking comparison of a square trajectory with the INDI and ANDI controllers.

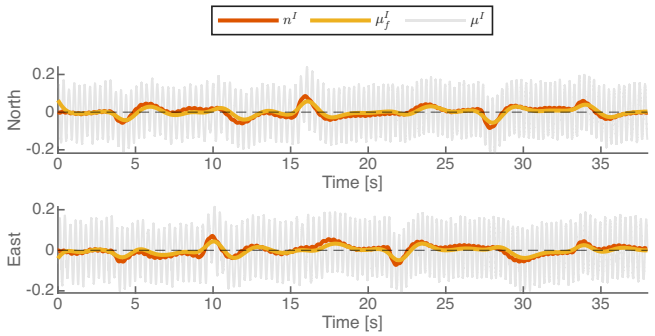


Figure 11: Principal axis of rotation tracking with the INDI controller.

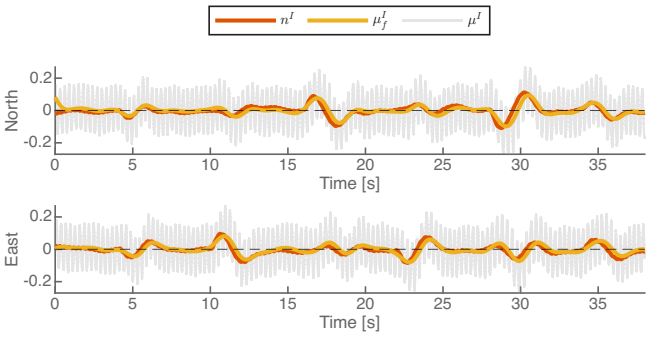


Figure 12: Principal axis of rotation tracking with the ANDI controller.

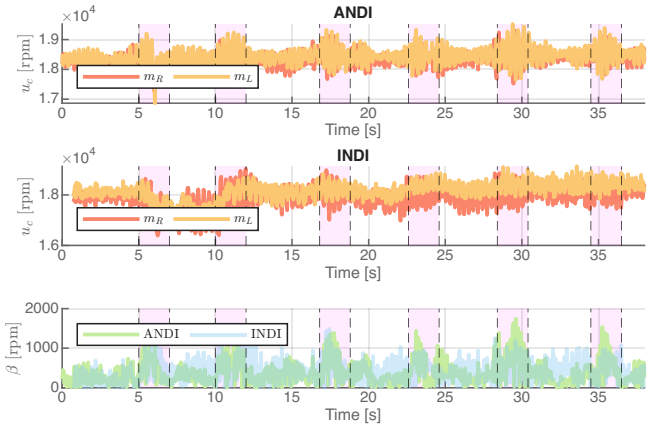


Figure 13: Comparison of control effort during the square tracking task between the INDI and ANDI controllers. Shaded areas indicate time intervals in which the drone is actively moving.

rate tracking of the desired inertial direction $\boldsymbol{n}^I$ using the projection of the principal axis of rotation onto the inertial frame, $\boldsymbol{\mu}^I$. However, while both controllers appear to perform very similarly in this task, there are subtle differences between them, which are elucidated in Fig. 13.

The commanded signals to the left and right motors during the square tracking test for both the INDI and ANDI controllers are presented in Figure 13. It also illustrates the variance of the control effort for the actuated axis, which is calculated as:

$$\hat{u}_c = u_{c_{m_L}} - u_{c_{m_R}}, \quad \beta = \left\| \hat{u}_c - \underline{\hat{u}_c} \right\|, \tag{38}$$

where $\underline{\hat{u}_c}$ denotes the moving average of $\hat{u}_c$. Thus, β serves as a measure of how much the control effort oscillates around its mean.

The results in Figure 13 show that during the intervals when the drone is actively moving and accelerating, the required control effort is generally higher for ANDI than for INDI. This behavior is expected, as the gyroscopic effect naturally opposes changes to the principal axis of rotation. Because ANDI explicitly compensates for this effect in advance, it computes a higher commanded effort to force the required motion.

Interestingly, however, during the intervals when the drone is stationary at the vertices of the square, where the controller is simply regulating its position and rejecting disturbances, the control effort is instead higher for INDI. As established in Section 4, at a yaw rate of $20\,\mathrm{rad\,s^{-1}}$, the control bandwidth of the angular acceleration in INDI is already degraded. This higher control effort indicates poorer control allocation; even when the drone is nominally hovering in place, INDI must continuously compensate for its own allocation inaccuracies, resulting in greater overall actuator activity.

Finally, ANDI was tested on a square trajectory above INDI's maximum stable spin rate of $33.82\,\mathrm{rad\,s^{-1}}$. Figures 14 and 15 confirm that ANDI successfully maintains accurate tracking at the yaw rate of roughly $34.5\mathrm{rad\,s^{-1}}$.

http://www.imavs.org/

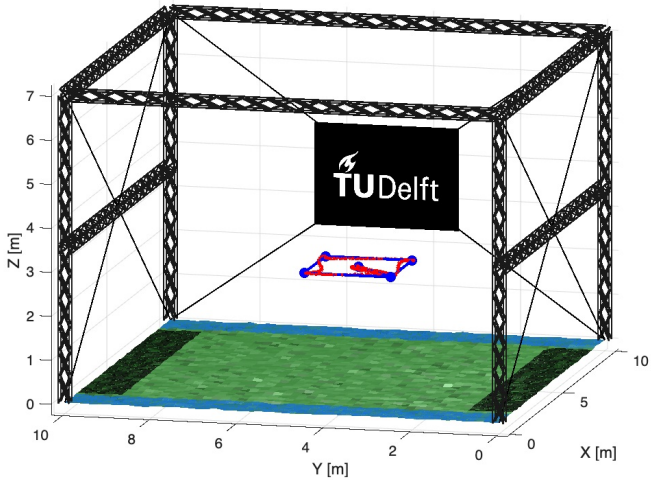


Figure 14: 3D flight path of the square trajectory using the ANDI controller at an average spin rate of $34.5\,\mathrm{rad\,s^{-1}}$ recorded at the Cyberzoo.

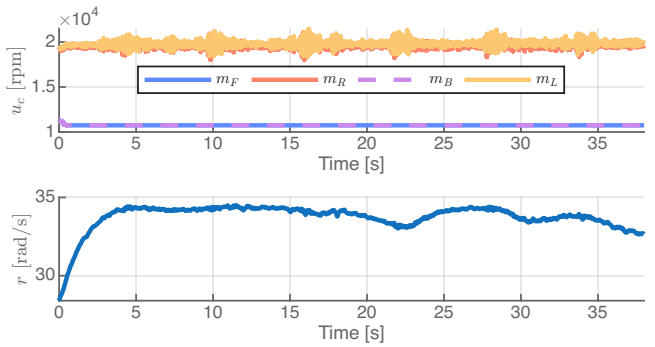


Figure 15: Command signals and yaw rate during the ANDI square tracking test at high spin rate.

8 Conclusion

This paper has demonstrated that the proposed ANDI controller can successfully guide and stabilize a quadcopter experiencing a complete loss of control authority in two opposing motors. The proposed framework was theoretically analyzed, verified in simulation, and validated through real-world flight tests, drawing a direct performance comparison with a comparable INDI architecture previously documented in the literature.

The results demonstrate that by explicitly incorporating knowledge of actuator dynamics and the kinematics of a spinning body into the control allocation, the ANDI controller successfully compensates for state-induced gyroscopic effects. Consequently, the ANDI framework is capable of maintaining stable flight and accurate trajectory tracking at significantly higher spin rates than INDI, which suffers from progressive allocation degradation under identical conditions. For platforms characterized by high required spin rates and relatively slow actuators, this extended operational range marks the difference between successfully executing a relaxed-hover solution and experiencing a total loss of control. An additional advantage of the ANDI controller is observed in its efficiency during stationary flight; the ANDI framework exhibits a lower overall control effort variance compared to INDI, indicating superior control allocation accuracy and a reduction in unnecessary actuator compensation.

Future work should focus on a rigorous robustness analysis to determine the extent to which the ANDI controller retains its performance advantages under severe model inaccuracies, parametric uncertainties, and unmodeled delays.

Acknowledgements

This research was co-funded by the Royal Dutch Navy and the Dutch Coast Guard. This initiative aims to foster collaboration and innovation in maritime endeavors.

References

[1] Mark W. Mueller and Raffaello D'Andrea. Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers. In *2014International Conference on Robotics and Automation (ICRA)*, pages 45–52, Hong Kong, China, May 2014. IEEE.

[2] Mark W. Mueller and Raffaello D'Andrea. Relaxed hover solutions for multicopters: Application to algorithmic redundancy and novel vehicles. *The International Journal of Robotics Research*, 35(8):873–889, 2016.

[3] Sihao Sun, Leon Sijbers, Xuerui Wang, and Coen de Visser. High-Speed Flight of Quadrotor Despite Loss of Single Rotor. *IEEE Robotics and Automation Letters*, 3(4):3201–3207, October 2018.

[4] Sihao Sun, Xuerui Wang, Qiping Chu, and Coen de Visser. Incremental Nonlinear Fault-Tolerant Control of a Quadrotor With Complete Loss of Two Opposing Rotors. *IEEE Transactions on Robotics*, 37(1):116–130, February 2021.

[5] 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, April 2018.

[6] Ole Pfeifle and Walter Fichter. Cascaded Incremental Nonlinear Dynamic Inversion for Three-Dimensional Spline-Tracking with Wind Compensation. *Journal of Guidance, Control, and Dynamics*, 44(8):1559–1571, August 2021.

[7] Tomaso M. L. De Ponti, Ewoud J. J. Smeur, and B. W. D. Remes. Incremental Nonlinear Dynamic Inversion controller for a Variable Skew Quad Plane. In

http://www.imavs.org/

2023 International Conference on Unmanned Aircraft Systems (ICUAS), pages 241–248, June 2023.

[8] P. Simplício, M. D. Pavel, E. van Kampen, and Q. P. Chu. An acceleration measurements-based approach for helicopter nonlinear flight control using Incremental Nonlinear Dynamic Inversion. *Control Engineering Practice*, 21(8):1065–1077, August 2013.

[9] Ewoud J.J. Smeur, Guido C.H.E. De Croon, and Qiping Chu. Gust disturbance alleviation with Incremental Nonlinear Dynamic Inversion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5626–5631, October 2016.

[10] Ewoud J. J. Smeur, Qiping Chu, and Guido C. H. E. de Croon. Adaptive Incremental Nonlinear Dynamic Inversion for Attitude Control of Micro Air Vehicles. *Journal of Guidance, Control, and Dynamics*, 39(3):450–461, March 2016.

[11] Tomaso M. L. De Ponti, Ewoud J. J. Smeur, and B. W. D. Remes. Incremental nonlinear fault-tolerant control of the variable skew quad plane with loss of two opposing rotors. In *2026 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 241–248, June 2026.

[12] S. Sieberling, Q. P. Chu, and J. A. Mulder. Robust Flight Control Using Incremental Nonlinear Dynamic Inversion and Angular Acceleration Prediction. *Journal of Guidance, Control, and Dynamics*, 33(6):1732–1742, November 2010.

[13] Stefan A. Raab, Jiannan Zhang, Pranav Bhardwaj, and Florian Holzapfel. Consideration of Control Effector Dynamics and Saturations in an Extended INDI Approach. In *AIAA Aviation 2019 Forum*. June 2019.

[14] Agnes Steinert, Stefan Raab, Simon Hafner, Florian Holzapfel, and Haichao Hong. From fundamentals to applications of incremental nonlinear dynamic inversion: A survey on INDI – Part I. *Chinese Journal of Aeronautics*, April 2025.

[15] Pranav Bhardwaj, Stefan A. Raab, and Florian Holzapfel. Higher Order Reference Model for Continuous Dynamic Inversion Control. In *AIAA Scitech 2021 Forum*, January 2021.

[16] Rasmus Steffensen, Agnes Steinert, and Ewoud J. J. Smeur. Nonlinear Dynamic Inversion with Actuator Dynamics: An Incremental Control Perspective. *Journal of Guidance, Control, and Dynamics*, 46(4):709–717, 2023.

[17] Yue Liu, Agnes Steinert, Haichao Hong, Hangxu Li, and Shiqiang Hu. Nonlinear dynamic inversion with generalized actuator dynamics: Performance and stability. *Aerospace Science and Technology*, 162:110218, July 2025.

[18] Tomaso Maria Luigi De Ponti, Ewoud Jan Jacob Smeur, and Bart Diane Walter Remes. Unified-actuator nonlinear dynamic inversion controller for the variable skew quad plane. *Journal of Guidance, Control, and Dynamics*, 48(5):1167–1176, 2025.

[19] Tomaso De Ponti, Simon Hafner, Ewoud Smeur, and Bart Remes. Actuator nonlinear dynamic inversion for the variable skew quad plane with heterogeneous motor dynamics. *Control Engineering Practice*, 171:106858, 2026.

[20] Justin P. G. Dubois, Evangelos Ntouros, and Ewoud J. J. Smeur. Full actuator nonlinear dynamic inversion for enhanced hybrid uav control. In *2026 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 241–248, June 2026.

[21] Ola Härkegård. Efficient active set algorithms for solving constrained least squares problems in aircraft control allocation. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002*, pages 1295–1300. IEEE, 2002.

[22] Ewoud Smeur, Daan Höppener, and Christophe De Wagter. Prioritized control allocation for quadrotors subject to saturation. In *International Micro Air Vehicle Conference and Flight Competition 2017*, pages 37–43, 2017.

[23] Tomaso De Ponti. Test data of actuator nonlinear fault-tolerant control of a fast-spinning quadrotor with loss of two opposing rotors, 2026.

http://www.imavs.org/

AERO-LQG: Aerial-Enabled Robust Optimization for LQG-Based Quadrotor Flight Controller

Daniel Engelsman and Itzik Klein*
Hatter Department of Marine Technologies, University of Haifa, Israel.

http://www.imavs.org/

Abstract

Quadrotors are vital across civilian, industrial, and military domains, where balancing the high power demands of agility with energy efficiency remains a critical constraint. Meeting this trade-off requires mode-specific optimization frameworks. While optimal control policies offer robust performance, selecting their underlying weight matrices lacks analytical guidance and typically relies on exhaustive search. This interdependence can be framed as a bi-level optimization problem, where the outer loop determines weights a priori. This work introduces Aerial-Enabled Robust Optimization for LQG tuning (AERO-LQG), a framework employing evolutionary strategies to fine-tune Linear-Quadratic-Gaussian weighting parameters. Applied to the linearized hovering mode of quadrotor flight, AERO-LQG achieves performance gains of several tens of percent, underscoring its potential for high-performance, energy-efficient control. The project is available @ GitHub.

1 Introduction

Unmanned aerial vehicles (UAVs), particularly quadrotors, have rapidly evolved from research prototypes into practical tools across a broad range of applications. Their versatility and agility enable complex missions—such as surveillance [1], inspection [2], delivery [3], and search and rescue [4]—making them valuable assets in both civilian and military domains [5]. From a dynamical systems perspective, quadrotor flight modes span a wide stability spectrum—from near-equilibrium hovering to highly nonlinear, aggressive maneuvers. As a result, no single control strategy can achieve optimal performance across all regimes. Transitions between control modes are therefore common, especially as energy constraints impose competing demands: controllers must either deliver agility [6], or conversely, limit maneuverability to conserve energy and extend flight time [7, 8]. Monolithic control lacks such flexibility; adaptive scheduling with task-dependent adjustments is needed [9–11]. Frequent mode switching, however, complicates state estimation, which remains fragile to the quadrotor's nonlinear and underactuated dynamics—despite advances in hardware, inertial sensing, and AI-driven perception [12–18]. To mitigate these issues, linear quadratic Gaussian (LQG) control remains a staple in aerial robotics, valued for its theoretical rigor and optimality under specific assumptions [19]. Many stabilized flight modes can be approximated as linear systems, allowing precise LQG control, provided operating conditions stay near the linearization point [20]. Quadrotors are no exception—their hovering mode, in particular, can be effectively linearized to facilitate both control and state estimation. However, while the quadratic form of the cost function is convex for fixed weights, tuning these values constitutes an additional outer optimization task—highly non-convex due to the coupled estimator–controller dynamics [14]. This work investigates whether a bi-level optimization framework can systematically maximize the stable flight envelope of LQG-controlled quadrotors under dynamic state uncertainty.

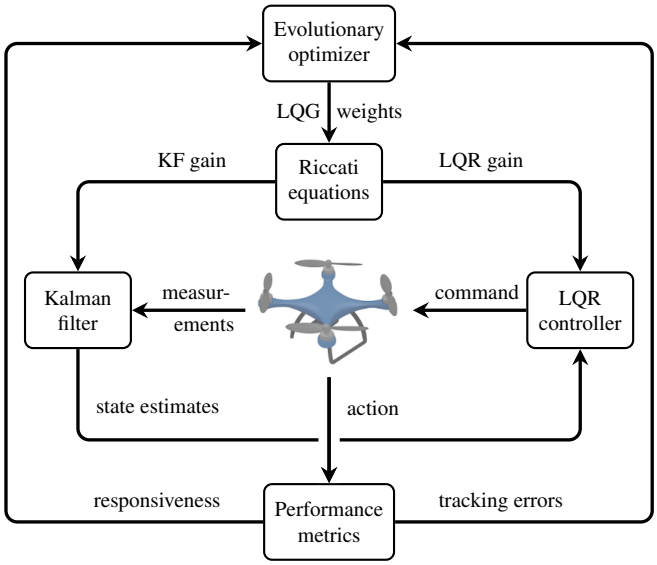


Figure 1: Flowchart of the AERO-LQG bi-level optimization.

To address this objective, we develop an aerial-enabled robust optimization for LQG tuning (AERO-LQG), tailored to modern quadrotors. As illustrated in Figure 1, an outer evolutionary loop refines the weighting matrices, while an inner loop computes the LQG gains, together forming a nested architecture. Evaluated across diverse metrics, AERO-LQG delivers unprecedented gains—reducing tracking error by over 55% and extending flight endurance by nearly 8%.

*Email addresses: {dengelsm@campus, kitzik@univ}.haifa.ac.il

http://www.imavs.org/

The paper unfolds as follows: Section 2 presents the problem setup and technical foundations; Section 3 outlines the proposed approach; Section 4 discusses the findings; and Section 5 concludes the study.

2 Problem Setup

Quadrotor behavior is modeled as a continuous-time dynamical system, defined by the following differential equation

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \,, \tag{1}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^n$ is the state vector, $\boldsymbol{u}(t) \in \mathbb{R}^m$ is the control input vector, and $\boldsymbol{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a smooth function that defines its dynamics. The solution to this system describes the trajectory of the state $\boldsymbol{x}(t)$, starting from an initial condition $\boldsymbol{x}(0)$ and evolving under the influence of the input $\boldsymbol{u}(t)$. Most aerial flight modes such as take-off, uncoordinated turns, maneuvers, and landing, require the system states to change dramatically, i.e. $\dot{\boldsymbol{x}}(t) \neq \boldsymbol{0}$, thereby causing $\boldsymbol{f}$ to evolve nonlinearly in time. That said, certain operating points, if controlled properly, enable maintaining the system at an equilibrium, denoted by subscript e, with their states demonstrating negligible change [21], such that

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}_e, \boldsymbol{u}_e) = \boldsymbol{0} \,. \tag{2}$$

The equilibrium behavior is assessed using local linearity, approximating the nonlinear dynamics $\boldsymbol{f}$ with a linear model near the equilibrium point $(\boldsymbol{x}_e, \boldsymbol{u}_e)$. General fixed-wing configurations permit linear approximation of several trimmed (stable) flight modes—such as straight-and-level flight, coordinated turns, and steady climbs or descents—thanks to the gliding capability of their large wing area. Rotorcraft, by contrast, depend entirely on rapid, coordinated motor corrections, yielding predominantly nonlinear dynamics [22, 23]. Consequently, the only flight mode that consistently maintains equilibrium under small perturbations is hovering (center), as depicted in the finite state machine in Figure 2.

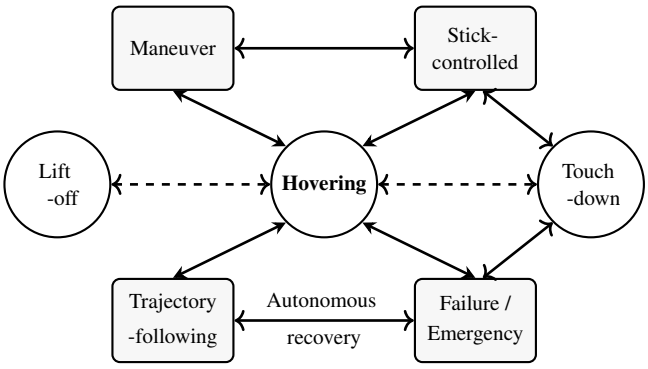


Figure 2: State transition diagram with hovering as the central hub connecting diverse nonlinear flight modes (boxes).

2.1 Equilibrium Point

Based on linear systems theory, steady-state dynamics allow the nonlinear system (1) to be linearized around the equilibrium point $(\boldsymbol{x}_e, \boldsymbol{u}_e)$ through a first-order Taylor expansion [24]. Since by definition, quadrotor hovering balances the net forces and net moments into zero, the following approximation holds around the hovering equilibrium point

$$\dot{\boldsymbol{x}}(t) \approx \mathbf{A}(\boldsymbol{x}_e, \boldsymbol{u}_e)(\boldsymbol{x}(t) - \boldsymbol{x}_e) + \mathbf{B}(\boldsymbol{x}_e, \boldsymbol{u}_e)(\boldsymbol{u}(t) - \boldsymbol{u}_e), \tag{3}$$

where $\boldsymbol{x}(t)$ is the state vector, $\boldsymbol{u}(t)$ the control inputs, and the deviations from equilibrium are expressed as

$$\delta\boldsymbol{x}(t) = \boldsymbol{x}(t) - \boldsymbol{x}_e \;\text{ and }\; \delta\boldsymbol{u}(t) = \boldsymbol{u}(t) - \boldsymbol{u}_e \,. \tag{4}$$

Assuming linear time-invariant (LTI) dynamics in the vicinity of the equilibrium point, the system Jacobians with respect to the state and input are defined as

$$\mathbf{A}(\boldsymbol{x}_e, \boldsymbol{u}_e) = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{(\boldsymbol{x}_e, \boldsymbol{u}_e)} \text{ and } \mathbf{B}(\boldsymbol{x}_e, \boldsymbol{u}_e) = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right|_{(\boldsymbol{x}_e, \boldsymbol{u}_e)} , \tag{5}$$

reducing the error dynamics to the standard LTI form

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \mathbf{A}\,\boldsymbol{x}(t) + \mathbf{B}\,\boldsymbol{u}(t) \,, \\ \boldsymbol{y}(t) &= \mathbf{C}\,\boldsymbol{x}(t) \,, \end{aligned} \tag{6}$$

where matrix $\mathbf{C}$ extracts the measurable states from $\boldsymbol{x}$. Model discrepancies and unmodeled dynamics are represented by zero-mean white Gaussian noise processes, with process covariance $\mathbf{W}$ and measurement covariance $\mathbf{V}$

$$\boldsymbol{w}(t) \sim \mathcal{N}(\boldsymbol{0}, \mathbf{W}) \,, \quad \mathbb{E}[\boldsymbol{w}\boldsymbol{w}^\top] = \mathbf{W} \succ 0 \,, \tag{7}$$

$$\boldsymbol{v}(t) \sim \mathcal{N}(\boldsymbol{0}, \mathbf{V}) \,, \quad \mathbb{E}[\boldsymbol{v}\boldsymbol{v}^\top] = \mathbf{V} \succeq 0 \,. \tag{8}$$

These propagate the open-loop uncertainty, whose state covariance $\mathbf{P}(t)$ evolves by the continuous-time Lyapunov equation

$$\dot{\mathbf{P}}(t) = \mathbf{A}\mathbf{P}(t) + \mathbf{P}(t)\mathbf{A}^\top + \mathbf{W}. \tag{9}$$

At any fixed time t, the joint distribution of the states and outputs can thus be approximated as Gaussian, namely

$$\begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{y}(t) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}(t) \\ \mathbf{C}\boldsymbol{\mu}(t) \end{bmatrix}, \begin{bmatrix} \mathbf{P}(t) & \mathbf{P}(t)\mathbf{C}^\top \\ \mathbf{C}\mathbf{P}(t) & \mathbf{C}\mathbf{P}(t)\mathbf{C}^\top + \mathbf{V} \end{bmatrix} \right). \tag{10}$$

where $\boldsymbol{\mu}(t)$ is the nominal noise-free state trajectory, initialized as $\boldsymbol{\mu}(t_0) = \mathbb{E}[\boldsymbol{x}(t_0)]$, with time evolution given by

$$\dot{\boldsymbol{\mu}}(t) = \mathbf{A}\boldsymbol{\mu}(t) + \mathbf{B}\boldsymbol{u}(t) \,. \tag{11}$$

http://www.imavs.org/

2.2 Inner-Level Optimization

Having identified the hovering operating point as locally linear under small perturbations enables the application of linear control theory. In this framework, solving the Riccati equations yields optimal controller ($\mathbf{K}$) and estimator ($\mathbf{L}$) gains, whereby the Kalman filter provides state estimates to the linear quadratic regulator (LQR). These gains are obtained by minimizing a quadratic cost over a finite time horizon T,

$$\min_{\boldsymbol{u}(t)} \mathcal{J}_{\text{in}} \triangleq \lim_{T\to\infty} \mathbb{E}\left[\frac{1}{T}\int_0^T \left(\|\boldsymbol{x}(t)\|^2_{\mathbf{Q}} + \|\boldsymbol{u}(t)\|^2_{\mathbf{R}}\right) \mathrm{d}t\right], \tag{12}$$

subject to (11), with a-priori specified $\mathbf{Q} \succeq 0$ and $\mathbf{R} \succ 0$ weighting deviations in $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$, respectively [25, 26]. By defining the estimation and the control errors as

$$\boldsymbol{e}(t) = \boldsymbol{x}(t) - \hat{\boldsymbol{x}}(t)\,, \tag{13}$$
$$\hat{\boldsymbol{\varepsilon}}(t) = \hat{\boldsymbol{x}}(t) - \boldsymbol{x}_{\text{ref}}\,, \tag{14}$$

the plant-estimator dynamics admits the block-state form of

$$\begin{bmatrix}\dot{\boldsymbol{x}}\\ \dot{\boldsymbol{e}}\end{bmatrix} = \begin{bmatrix}\mathbf{A}-\mathbf{BK} & \mathbf{BK}\\ \mathbf{0} & \mathbf{A}-\mathbf{LC}\end{bmatrix}\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{e}\end{bmatrix} + \begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{I} & -\mathbf{L}\end{bmatrix}\begin{bmatrix}\boldsymbol{w}\\ \boldsymbol{v}\end{bmatrix}, \tag{15}$$

highlighting the separation principle, with an upper-triangular form that decouples the estimator dynamics [27, 28].

2.3 System Dynamics

We begin by outlining the assumptions and coordinate frames relevant to hovering, a near-equilibrium condition that justifies standard modeling practices [29, 30]: (i) the quadrotor is a rigid symmetric body; (ii) thrust and moments scale with the square of rotor speeds; (iii) aerodynamic drag is neglected due to low speed; (iv) the center of mass coincides with the body-frame origin. The orientation of the body frame $\mathcal{B}$ relative to the inertial frame $\mathcal{I}$ is given by the rotation matrix

$$\mathcal{R}^{\mathcal{I}}_{\mathcal{B}} := \left(\mathbf{e}^{\mathcal{B}}_x, \mathbf{e}^{\mathcal{B}}_y, \mathbf{e}^{\mathcal{B}}_z\right) \in SO(3)\,, \tag{16}$$

which is constructed through a sequence of three consecutive rotations: $\mathcal{R}_z(\psi)$ about the $\mathbf{e}^{\mathcal{I}}_z$ axis, $\mathcal{R}_y(\theta)$ about the $\mathbf{e}^{\mathcal{I}}_y$ axis, and $\mathcal{R}_x(\phi)$ about the $\mathbf{e}^{\mathcal{I}}_x$ axis, where $\{\phi,\theta\} \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$ and $\psi \in (-\pi, \pi)$ correspond to the roll, pitch, and yaw angles, respectively. Based on the Newton-Euler formalism [31], the translational-rotational coupled dynamics are expressed as

$$\begin{pmatrix}\boldsymbol{f}^{\mathcal{B}}\\ \boldsymbol{\tau}^{\mathcal{B}}\end{pmatrix} = \begin{pmatrix}m\boldsymbol{I}_3 & \mathbf{0}_3\\ \mathbf{0}_3 & \boldsymbol{J}\end{pmatrix}\begin{pmatrix}\dot{\boldsymbol{v}}^{\mathcal{B}}\\ \dot{\boldsymbol{\omega}}^{\mathcal{B}}\end{pmatrix} + \begin{pmatrix}\boldsymbol{\omega}^{\mathcal{B}} \times m\,\boldsymbol{v}^{\mathcal{B}}\\ \boldsymbol{\omega}^{\mathcal{B}} \times \boldsymbol{J}\boldsymbol{\omega}^{\mathcal{B}}\end{pmatrix}, \tag{17}$$

where, m is the body mass; $\boldsymbol{J}$, $\boldsymbol{I}_3$, and $\mathbf{0}_3$ are 3×3 inertia tensor, identity, and zero matrices. Forces $\boldsymbol{f}$ and moments $\boldsymbol{\tau}$, expressed in the body frame, govern the time derivatives of $\boldsymbol{v}^{\mathcal{B}}$ and $\boldsymbol{\omega}^{\mathcal{B}}$. In fixed-point hover, near-zero body velocities make drag negligible, so the net body-frame force includes only gravity $\mathbf{g}^{\mathcal{B}} = \mathrm{g}\,\mathbf{e}^{\mathcal{B}}_z$ and vertical rotor thrusts $\boldsymbol{f}^{\mathcal{B}}_{z,i}$, as in

$$\boldsymbol{f}^{\mathcal{B}} = \sum_{i=1}^{4} \boldsymbol{f}^{\mathcal{B}}_{z,i} - m\,\boldsymbol{g}^{\mathcal{B}}\,. \tag{18}$$

Each rotor spins along the body-frame z-axis, generating thrust by the i-th propeller modeled as

$$\boldsymbol{f}^{\mathcal{B}}_{z,i} = f_{z,i}\,\mathbf{e}^{\mathcal{B}}_z = \left(k_T\Omega_i^2\right)\mathbf{e}^{\mathcal{B}}_z\,, \tag{19}$$

where k_T is a lumped coefficient. Attitude stabilization is achieved through differential thrust between opposing rotors spaced by ℓ. The rolling moment arises from the thrust difference between motors #4 (left) and #2 (right), given by

$$\boldsymbol{\tau}^{\mathcal{B}}_{\phi} = \ell\left(\boldsymbol{f}^{\mathcal{B}}_{z,4} - \boldsymbol{f}^{\mathcal{B}}_{z,2}\right)\mathbf{e}^{\mathcal{B}}_x\,, \tag{20}$$

while the pitch moment is controlled by the longitudinal thrust difference between motors #1 (front) and #3 (rear), as

$$\boldsymbol{\tau}^{\mathcal{B}}_{\theta} = \ell\left(\boldsymbol{f}^{\mathcal{B}}_{z,3} - \boldsymbol{f}^{\mathcal{B}}_{z,1}\right)\mathbf{e}^{\mathcal{B}}_y\,. \tag{21}$$

Finally, yaw arises from asymmetry in rotor drag torques, i.e., when $\sum_i \tau_{M,i} \neq 0$. Defining k_M as the torque coefficient and positive yaw as counter-clockwise about $\mathbf{e}^{\mathcal{B}}_z$, yaw torque is

$$\boldsymbol{\tau}^{\mathcal{B}}_{\psi} = \sum_{i=1}^{4} \tau_{M,i} = (-1)^i\left(k_M\Omega_i^2\right)\mathbf{e}^{\mathcal{B}}_z\,. \tag{22}$$

The platform's trajectory is derived by time-integrating its dynamics and applying a coordinate transformation to an inertial reference frame, letting the position vector be [19]

$$\boldsymbol{\xi}^{\mathcal{I}} = (x, y, z)^{\top}\,, \tag{23}$$

with the corresponding body velocities related by

$$\dot{\boldsymbol{\xi}}^{\mathcal{I}} = \mathcal{R}^{\mathcal{I}}_{\mathcal{B}}\boldsymbol{v}^{\mathcal{B}}\,. \tag{24}$$

The Euler angles vector in the inertial frame is denoted as

$$\boldsymbol{\eta}^{\mathcal{I}} = (\phi, \theta, \psi)^{\top}\,, \tag{25}$$

with its rate of change related to the body angular rates (p, q, r) through the transformation matrix $\boldsymbol{W}_{\boldsymbol{\eta}}$, such that

$$\dot{\boldsymbol{\eta}}^{\mathcal{I}} = \boldsymbol{W}_{\boldsymbol{\eta}}\,\boldsymbol{\omega}^{\mathcal{B}} = \begin{pmatrix}1 & \mathrm{s}_\phi\mathrm{t}_\theta & \mathrm{c}_\phi\mathrm{t}_\theta\\ 0 & \mathrm{c}_\phi & -\mathrm{s}_\phi\\ 0 & \mathrm{s}_\phi/\mathrm{c}_\theta & \mathrm{c}_\phi/\mathrm{c}_\theta\end{pmatrix}\begin{pmatrix}p\\ q\\ r\end{pmatrix}, \tag{26}$$

where 't' is the tangent function. For derivation, thrust and torques are idealized as direct control input functions, hence

$$\boldsymbol{u}(\Omega) = \begin{pmatrix}\boldsymbol{u}_z\\ \boldsymbol{u}_\phi\\ \boldsymbol{u}_\theta\\ \boldsymbol{u}_\psi\end{pmatrix} = \begin{pmatrix}f_z\,\mathbf{e}^{\mathcal{B}}_z\\ \tau_\phi\,\mathbf{e}^{\mathcal{B}}_x\\ \tau_\theta\,\mathbf{e}^{\mathcal{B}}_y\\ \tau_\psi\,\mathbf{e}^{\mathcal{B}}_z\end{pmatrix} \in \mathbb{R}^4\,. \tag{27}$$

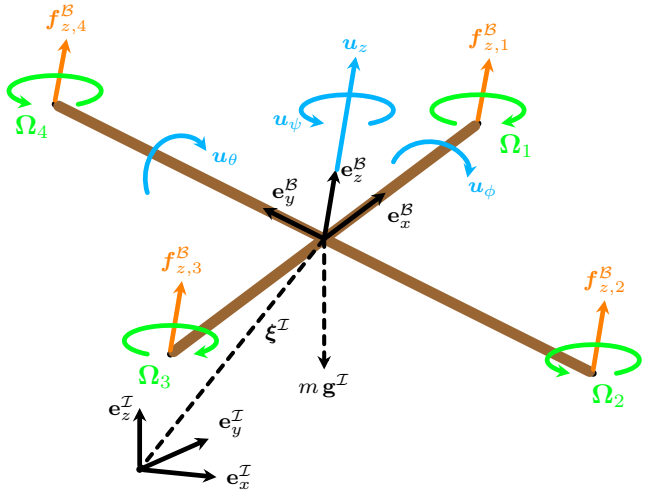


Figure 3: Free-body diagram of the quadrotor system model.

Finally, by incorporating the generalized coordinates into the system model, the augmented state vector is defined as

$$\boldsymbol{x} = \begin{pmatrix} (\boldsymbol{\xi}^{\mathcal{I}})^\top & (\boldsymbol{v}^{\mathcal{B}})^\top & (\boldsymbol{\eta}^{\mathcal{I}})^\top & (\boldsymbol{\omega}^{\mathcal{B}})^\top \end{pmatrix}^\top \in \mathbb{R}^{12}, \quad (28)$$

with the nonlinear state-space model in (1) evolves as

$$\begin{pmatrix} \dot{\boldsymbol{\xi}}^{\mathcal{I}} \\ \dot{\boldsymbol{v}}^{\mathcal{B}} \\ \dot{\boldsymbol{\eta}}^{\mathcal{I}} \\ \dot{\boldsymbol{\omega}}^{\mathcal{B}} \end{pmatrix} = \begin{pmatrix} (\mathcal{R}^{\mathcal{I}}_{\mathcal{B}})\,\boldsymbol{v}^{\mathcal{B}} \\ m^{-1}\boldsymbol{u}_z - \boldsymbol{\omega}^{\mathcal{B}} \times \boldsymbol{v}^{\mathcal{B}} - g\,\mathbf{e}^{\mathcal{B}}_z \\ (\boldsymbol{W}_{\boldsymbol{\eta}})\boldsymbol{\omega}^{\mathcal{B}} \\ \boldsymbol{J}^{-1}\left((\boldsymbol{u}_\phi + \boldsymbol{u}_\theta + \boldsymbol{u}_\psi) - \boldsymbol{\omega}^{\mathcal{B}} \times \boldsymbol{J}\boldsymbol{\omega}^{\mathcal{B}}\right) \end{pmatrix}. \quad (29)$$

Figure 3 depicts the quadrotor free-body diagram: airframe (brown), reference axes (black), rotor thrust vectors (green), control inputs (azure), and rotor spin actuation (orange); gravity and inertial displacement are shown as dashed guides.

$$\mathbf{B} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\Big|_{(\boldsymbol{x}_e, \boldsymbol{u}_e)} = \begin{pmatrix} \mathbf{0}_{3\times 1} & \mathbf{0}_{3\times 3} \\ m^{-1}\mathbf{e}^{\mathcal{B}}_z & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 1} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 1} & \boldsymbol{J}^{-1} \end{pmatrix} \in \mathbb{R}^{12\times 4}. \quad (30)$$

3 Proposed Approach

We begin by analyzing the linearized dynamics about the hover equilibrium, outlining the outer evolutionary LQG optimization process, and lastly presenting our proposed implementation amongst several state-of-the-art tuning methods.

3.1 Hovering Linearization

Since hovering occurs near equilibrium with small state variables, local behavior can be approximated using small-angle assumptions. This simplifies (26) to $\dot{\boldsymbol{\eta}}^{\mathcal{I}} \approx \boldsymbol{\omega}^{\mathcal{B}}$ and $\mathcal{R}^{\mathcal{I}}_{\mathcal{B}} \approx \boldsymbol{I}_3 + (\boldsymbol{\eta})_\times$, respectively, where $(\cdot)_\times$ denotes the skew-symmetric operator. With these approximations, the Jacobian of the nonlinear dynamics in (29) is computed as

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \begin{pmatrix} \mathbf{0}_3 & \boldsymbol{I}_3 + (\boldsymbol{\eta})_\times & -(\boldsymbol{v}^{\mathcal{B}})_\times & \mathbf{0}_3 \\ \mathbf{0}_3 & -(\boldsymbol{\omega}^{\mathcal{B}})_\times & -(\boldsymbol{g}^{\mathcal{I}})_\times & (\boldsymbol{v}^{\mathcal{B}})_\times \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{I}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \nabla_{\boldsymbol{\omega}}(\dot{\boldsymbol{\omega}}^{\mathcal{B}}) \end{pmatrix}, \quad (31)$$

with the corresponding sub-Jacobian of $\dot{\boldsymbol{\omega}}^{\mathcal{B}}$ given by

$$\nabla_{\boldsymbol{\omega}}(\dot{\boldsymbol{\omega}}^{\mathcal{B}}) = \boldsymbol{J}^{-1}\left((\boldsymbol{J}\boldsymbol{\omega}^{\mathcal{B}})_\times - \boldsymbol{\omega}^{\mathcal{B}} \times \boldsymbol{J}\right). \quad (32)$$

Assuming a hovering equilibrium defined by a state vector at a fixed point and a nominal thrust control input

$$\boldsymbol{x}_e = \begin{pmatrix} \boldsymbol{\xi}^{\mathcal{I}}_e & \mathbf{0}_3 & \boldsymbol{\eta}^{\mathcal{I}}_e & \mathbf{0}_3 \end{pmatrix}^\top \in \mathbb{R}^{12}, \quad (33)$$

$$\boldsymbol{u}_e = \begin{pmatrix} mg & 0 & 0 & 0 \end{pmatrix}^\top \in \mathbb{R}^4, \quad (34)$$

substituting this operating point $(\boldsymbol{x}_e, \boldsymbol{u}_e)$ into the state Jacobian yields the 12×12 linearized system dynamics model

$$\mathbf{A} = \begin{pmatrix} \mathbf{0}_3 & \boldsymbol{I}_3 + (\boldsymbol{\eta}_e)_\times & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & -(\boldsymbol{g}^{\mathcal{I}})_\times & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{I}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \end{pmatrix}, \quad (35)$$

where higher-order centrifugal terms and their Coriolis couplings vanish. Likewise, linearizing the control input around the thrust required to balance gravity gives Within the state estimation process, these linearized dynamics hold locally around the hover equilibrium $(\boldsymbol{x}_e, \boldsymbol{u}_e)$, enabling state propagation via the state-space model (6).

Table 1: Comparative analysis of weight optimization methods, ordered top-to-bottom by their novelty and precision.

Symbol	Optimization method	Approach	Pros	Cons	Complexity
MT	Manual tuning	Trial-and-error	Trivial implementation	Inefficient (exhaustive)	$O(n_p f_e)$
BR	Bryson's rule [32]	Heuristic	Intuitive; Fast initial guess	Inherently suboptimal	$O(f_e)$
PS	Particle swarm [33]	Swarm-based	Robust against local minima	Premature convergence	$O(k_i n_p f_e)$
GA	Genetic algorithm [34]	Evolutionary	Global search; gradient-free	Computation-intensive	$O(k_i n_p f_e)$
BY	Bayesian [35]	Surrogate-based	Sample-efficient	Scalability issues	$O(k_i f_e + k_i^3)$
CMA	Matrix adaptation [36]	Simulation-based	Robust in rugged search spaces	Slower convergence	$O(k_i n_p f_e)$

http://www.imavs.org/

http://www.imavs.org/

Table 1 lists the six optimization techniques used as benchmarks in this study, distinguished by their key characteristics and ranked in ascending order of complexity. To assess the effectiveness of our covariance matrix adaptation (CMA) implementation—highlighted in green at the bottom—complexity is expressed in terms of number of parameters (n_p), iterations (k_i), and cost per function evaluation (f_e).

3.2 *Outer-Level Optimization*

Gradient-based methods, while effective in high-dimensional settings, are unsuitable here since analytical gradients are unavailable: the cost depends implicitly on closed-loop dynamics. The LQG cost (12) is set a priori by $\mathbf{Q}$ and $\mathbf{R}$, whose iterative tuning forms an inner feedback loop within the outer optimizer. As hovering prioritizes position holding, followed by attitude and yaw stabilization, the outer cost is defined as

$$\mathcal{J}_{\text{out}} = \int_0^T \Big(\|\boldsymbol{\varepsilon}_{\boldsymbol{\xi}}(t)\|_2 + \lambda \|\boldsymbol{\varepsilon}_{\boldsymbol{\eta}}(t)\|_2 \Big) \, \mathrm{d}t \,, \tag{36}$$

where $\lambda \ll 1$ penalizes the translational–rotational error trade-off. For a state-space model (6) with n states and m control inputs, the search spans $n^2 + m^2$ weight parameters. This places the bi-level tuning firmly in the domain of black-box optimization, where weights are iteratively refined without explicit gradients or access to internal system dynamics. Algorithm 1 formalizes the proposed framework in a high-level, optimizer-agnostic form, with the inner LQG control loop nested within the outer weight-optimization process.

Algorithm 1: AERO-LQG bi-level optimization structure

Input: Optimizer $\mathcal{O}$, weight λ, tolerance ϵ
Initialize $\mathcal{O}$ and weights $(\mathbf{Q}, \mathbf{R})$; `// outer-loop`
while *not converged* **do**
  Propose candidate sets $\{(\mathbf{Q}_i, \mathbf{R}_i)\}$ using $\mathcal{O}$;
  foreach *candidate* $(\mathbf{Q}_i, \mathbf{R}_i)$ **do**
    $(\boldsymbol{\xi}(t), \boldsymbol{\eta}(t)) \leftarrow \arg\min \mathcal{J}_{\text{in}}$; `// inner-loop`
  Feed optimal errors $(\boldsymbol{\varepsilon}^*_{\boldsymbol{\xi}}, \boldsymbol{\varepsilon}^*_{\boldsymbol{\eta}})$ into optimizer $\mathcal{O}$;
  if $\mathcal{J}_{out} < \epsilon$ **then**
    break; `// convergence`
return $(\mathbf{Q}^*, \mathbf{R}^*)$;

4 ANALYSIS AND RESULTS

To explore the intricacies of LQG tuning in quadrotor hovering, AERO-LQG results are organized into three subsections.

4.1 *Cost Landscape*

When applied to a 12-D state space rich in cross-coupled dynamics, the LQG quadratic cost (12) yields a highly non-convex landscape over the gain parameters. In such settings, gradient-based optimizers are prone to becoming trapped in inferior local minima, with little prospect of escape.

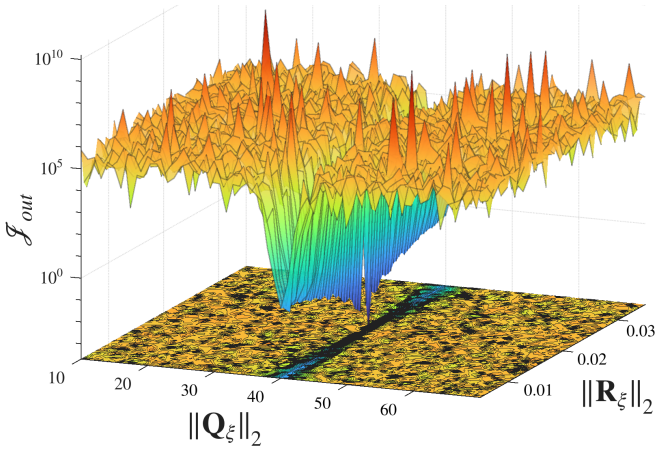


Figure 4: LQG's marginal stability spawns a rugged, minima-rich cost surface, calling for gradient-free optimization.

To demonstrate, we compute the cost of a 10-second standardized gust trajectory, assuming isotropic 3D errors. For clarity, $\mathbf{Q}_\xi$ and $\mathbf{R}_\xi$ utilize standard L_2 norms, while the estimator noise covariance matrices $\mathbf{W}$ and $\mathbf{V}$ remain fixed a priori. Figure 4 illustrates the scalar surface riddled with local minima, exposing the shortcomings of gradient-based techniques. This accentuates the challenge of proper LQG tuning: large regions of the cost surface correspond to poor controller performance, with log-scale costs on the order of $\mathcal{J}_{\text{out}} \gtrsim 10^5$. Yet, precise calibration of $\mathbf{Q}_\xi$ and $\mathbf{R}_\xi$ norms reduces the cost by nearly seven orders of magnitude, achieving near-global optimality and substantially improved performance.

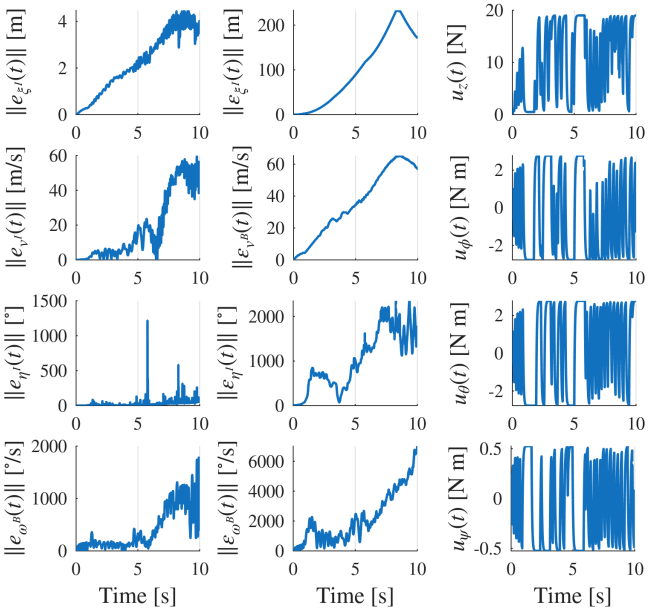


Figure 5: State trajectories under nominal LQG tuning.

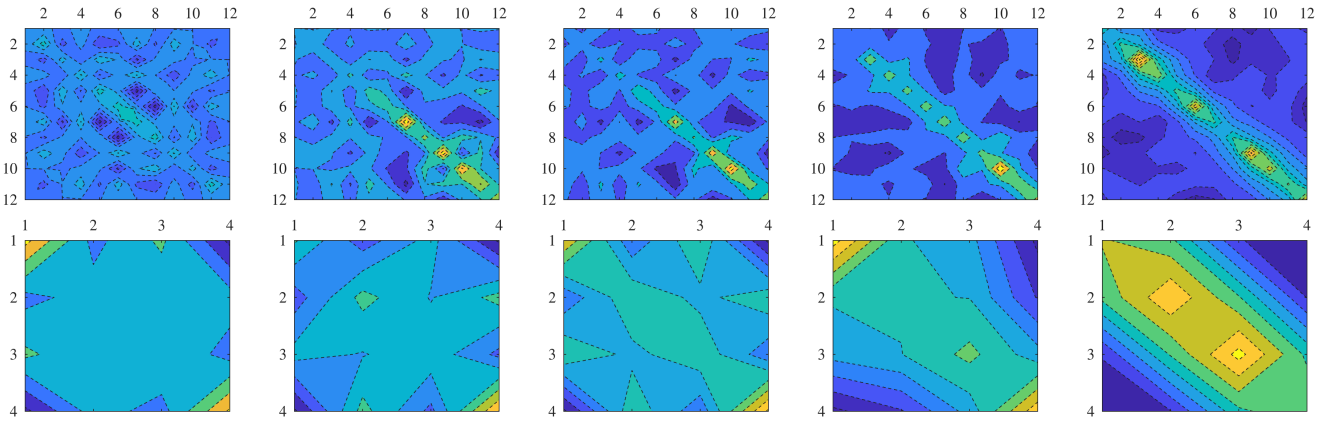


Figure 6: Evolutionary optimization guides $\mathbf{Q}$ (top) and $\mathbf{R}$ (bottom) from random start (left) toward diagonal dominance (right).

4.2 *Robustness Limits under Disturbances*

Figure 5 illustrates the quadrotor states over a 10-second gust impulse [7], with 3D error norms (13) and (14) on the left, and control inputs (27) on the right. Even with careful manual tuning, the system diverges. In delicate hovering, LQG's narrow stability margins make it highly sensitive to disturbances; once error assumptions are violated, divergence grows unchecked. The actuators respond with rapid but ineffective corrections, oscillating between their mechanical limits. This instability stems from the tight coupling between the estimator and controller: small deviations degrade the Kalman filter, initiating a feedback loop that amplifies errors. Unlike LQR, which acts directly on measured states, LQG's reliance on these imperfect state estimates compounds its sensitivity to inaccuracies.

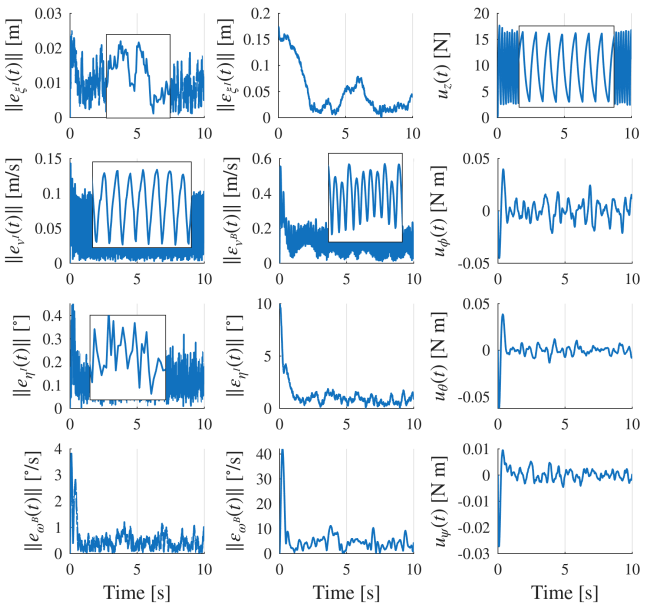


Figure 7: State trajectories under AERO-LQG tuning.

4.3 *Performance Evaluation*

Building on the need for robust, global optimization methods that withstand large perturbations, this section: (i) presents our CMA implementation, (ii) analyzes its performance, and (iii) benchmarks it against several state-of-the-art approaches. Figure 6 illustrates CMA's iterative refinement across five stages, guiding the search distribution from random initialization (left) to convergence (right). Normalized for clarity, both $\mathbf{Q}$ and $\mathbf{R}$ progressively concentrate weight along the diagonal as off-diagonal entries diminish, indicating negligible cross-correlation.

Figure 7 follows the same top-to-bottom error analysis under these finalized parameters. In contrast to the saturated controllers in Figure 5, the optimized trajectories here recover immediately from the same unit disturbance. A residual drift persists, visible as sawtooth patterns in the insets; however, the actuators remain within limits, achieving stabilization with significantly lower control effort (rightmost column). Table 2 provides supporting numbers: MT at the top exhibits complete divergence (∞), illustrating why manual tuning falls short next to automated algorithms.

Table 2: Comparison of optimizers by final-time errors.

Error / Model	Position [m]		Orientation [deg]		Effort [N·s]
	$\|\boldsymbol{e}_{\boldsymbol{\xi}^{\mathcal{I}}}\|$	$\|\boldsymbol{\varepsilon}_{\boldsymbol{\xi}^{\mathcal{I}}}\|$	$\|\boldsymbol{e}_{\boldsymbol{\eta}^{\mathcal{I}}}\|$	$\|\boldsymbol{\varepsilon}_{\boldsymbol{\eta}^{\mathcal{I}}}\|$	$\int_0^T \boldsymbol{u}(t)\,\mathrm{d}t$
MT	$\to\infty$	$\to\infty$	$\to\infty$	$\to\infty$	$\to\infty$
BR	0.48	1.12	0.57	12.3	$>10^3$
PS	0.22	0.77	0.18	6.13	293.8
GA	0.03	0.09	0.22	2.05	119.1
BY	0.31	0.27	0.47	3.21	143.1
CMA	**0.01**	**0.05**	**0.17**	**1.74**	**98.6**

http://www.imavs.org/

In contrast, the optimizers below perform far better, with CMA surpassing the leading GA by reducing estimation and control errors by at least 33% and 55%, respectively. Figure 8 shows these differences across eight flight metrics from [7], benchmarking all candidate optimizers (excluding MT) with color-coded distinction.

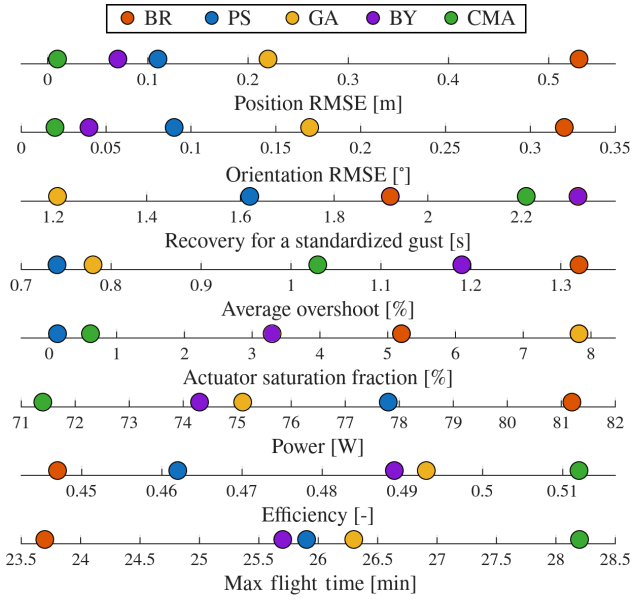


Figure 8: Comparative analysis of the candidate optimizers.

As shown, CMA (green) attains optimality in five of eight criteria, excelling in precision and energy efficiency but lagging in agility and responsiveness. Overall, evolutionary strategies (PS, GA, CMA) consistently outperform heuristic methods (MT, BR) and Bayesian optimization by avoiding premature convergence in dense local minima. These advantages provide tangible benefits, particularly in resource-constrained applications where extended flight endurance is critical.

5 Discussion and Conclusion

This work introduced AERO-LQG, a systematic weight-tuning framework for quadrotor hovering that addresses the narrow stability margins and highly non-convex cost landscapes characteristic of such systems. By decoupling the inner control loop from the outer stochastic optimizer, weight initialization is delegated to a powerful evolutionary strategy that remains robust to complex cost landscapes. Benchmarking across eight flight metrics showed the consistent superiority of our CMA implementation in key hovering capabilities—energy efficiency, flight endurance, and disturbance rejection—a result particularly significant for resource-constrained platforms. While demonstrated on an aerial system, the approach generalizes to other robots, offering resource-aware, high-performance control for real-world demands.

References

[1] Kevin Leahy, Dingjiang Zhou, Cristian-Ioan Vasile, Konstantinos Oikonomopoulos, Mac Schwager, and Calin Belta. Persistent surveillance for unmanned aerial vehicles subject to charging and temporal logic constraints. *Autonomous Robots*, 40(8):1363–1378, 2016.

[2] Han Liang, Seong-Cheol Lee, Woosung Bae, Jeongyun Kim, and Suyoung Seo. Towards UAVs in construction: advancements, challenges, and future directions for monitoring and inspection. *Drones*, 7(3):202, 2023.

[3] Xueping Li, Jose Tupayachi, Aliza Sharmin, and Madelaine Martinez Ferguson. Drone-aided delivery methods, challenge, and the future: A methodological review. *Drones*, 7(3):191, 2023.

[4] Mingyang Lyu, Yibo Zhao, Chao Huang, and Hailong Huang. Unmanned aerial vehicles for search and rescue: A survey. *Remote Sensing*, 15(13):3266, 2023.

[5] Michael W Lewis. Drones and the boundaries of the battlefield. *Tex. Int'l LJ*, 47:293, 2011.

[6] Drew Hanover, Antonio Loquercio, Leonard Bauersfeld, Angel Romero, Robert Penicka, Yunlong Song, Giovanni Cioffi, Elia Kaufmann, and Davide Scaramuzza. Autonomous drone racing: A survey. *IEEE Transactions on Robotics*, 40:3044–3067, 2024.

[7] Daniel Engelsman and Itzik Klein. C-ZUPT: Stationarity-aided aerial hovering. *IEEE Transactions on Aerospace and Electronic Systems*, pages 1–15, 2026.

[8] Leonard Bauersfeld and Davide Scaramuzza. Range, endurance, and optimal speed estimates for multicopters. *IEEE Robotics and Automation Letters*, 7(2):2953–2960, 2022.

[9] Chun Fui Liew, Danielle DeLatte, Naoya Takeishi, and Takehisa Yairi. Recent developments in aerial robotics: A survey and prototypes overview. *arXiv preprint arXiv:1711.10085*, 2017.

[10] Nourhan Elmeseiry, Nancy Alshaer, and Tawfik Ismail. A detailed survey and future directions of unmanned aerial vehicles (UAVs) with potential applications. *Aerospace*, 8(12):363, 2021.

[11] Khaled Telli, Okba Kraa, Yassine Himeur, Abdelmalik Ouamane, Mohamed Boumehraz, Shadi Atalla, and Wathiq Mansoor. A comprehensive review of recent research trends on unmanned aerial vehicles (UAVs). *Systems*, 11(8):400, 2023.

[12] Naser El-Sheimy and You Li. Indoor navigation: State of the art and future trends. *Satellite Navigation*, 2(1):7, 2021.

http://www.imavs.org/

[13] Daniel Engelsman and Itzik Klein. Information-aided inertial navigation: A review. *IEEE Transactions on Instrumentation and Measurement*, 72:1–18, 2023.

[14] Alessandro Saviolo and Giuseppe Loianno. Learning quadrotor dynamics for precise, safe, and agile flight control. *Annual Reviews in Control*, 55:45–60, 2023.

[15] Nadav Cohen and Itzik Klein. Inertial navigation meets deep learning: A survey of current trends and future directions. *Results in Engineering*, page 103565, 2024.

[16] Daniel Engelsman and Itzik Klein. Towards learning-based gyrocompassing. *Engineering Applications of Artificial Intelligence*, 163:112842, 2026.

[17] Barak Diker and Itzik Klein. Neural aided adaptive innovation-based invariant kalman filter. *arXiv preprint arXiv:2603.26709*, 2026.

[18] Guy Damari, Zeev Yampolsky, Nadav Cohen, Arup Kumar Sahoo, Jeryes Danial, Felipe O Silva, and Itzik Klein. AI-aided advancements in autonomous underwater vehicle navigation. *arXiv preprint arXiv:2605.04672*, 2026.

[19] Kimon P Valavanis and George J Vachtsevanos. *Handbook of unmanned aerial vehicles*. Springer Publishing Company, Incorporated, 2014.

[20] Labane Chrif and Zemalache Meguenni Kadda. Aircraft control system using LQG and LQR controller with optimal estimation-Kalman filter design. *Procedia Engineering*, 80:245–257, 2014.

[21] David K Arrowsmith and Colin M Place. *An introduction to dynamical systems*. Cambridge university press, 1990.

[22] Amit Levy and Itzik Klein. Adaptive neural unscented Kalman filter. *IEEE Transactions on Intelligent Vehicles*, 11(3):428–437, 2026.

[23] Janis Peksa and Dmytro Mamchur. A review on the state of the art in copter drones and flight control systems. *Sensors*, 24(11):3349, 2024.

[24] Michael V Cook. *Flight dynamics principles: a linear systems approach to aircraft stability and control*. Butterworth-Heinemann, 2012.

[25] Karl J Åström. *Introduction to stochastic control theory*. Courier Corporation, 2012.

[26] Daniel Engelsman and Itzik Klein. Inertial-based LQG control: A new look at inverted-pendulum stabilization. *International Journal of Mechanical System Dynamics*, 2025.

[27] NI Mahmudov and A Denker. On controllability of linear stochastic systems. *International Journal of Control*, 73(2):144–151, 2000.

[28] Daniel Engelsman and Itzik Klein. Revisiting certainty equivalence: The structural coupling between estimation and control in underactuated nonlinear systems. *arXiv preprint arXiv:2607.07276*, 2026.

[29] Gabriel Hoffmann, Haomiao Huang, Steven Waslander, and Claire Tomlin. Quadrotor helicopter flight dynamics and control: Theory and experiment. In *AIAA guidance, navigation and control conference and exhibit*, page 6461, 2007.

[30] Sara Sadr, S Ali A Moosavian, and Payam Zarafshan. Dynamics modeling and control of a quadrotor with swing load. *Journal of Robotics*, 2014(1):265897, 2014.

[31] Pedro Castillo, Rogelio Lozano, and Alejandro Enrique Dzul. *Modelling and control of mini-flying machines*. Springer Science & Business Media, 2005.

[32] J.P. Hespanha. *Linear Systems Theory: Second Edition*. Princeton University Press, 2018.

[33] Rabii Fessi and Soufiene Bouallègue. LQG controller design for a quadrotor uav based on particle swarm optimisation. *International Journal of Automation and Control*, 13(5):569–594, 2019.

[34] Abdulla I Abdulla and Ibrahim K Mohammed. Aircraft pitch control design using LQG controller based on genetic algorithm. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 21(2):409–417, 2023.

[35] Alonso Marco, Philipp Hennig, Jeannette Bohg, Stefan Schaal, and Sebastian Trimpe. Automatic LQR tuning based on gaussian process global optimization. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 270–277. IEEE, 2016.

[36] Nikolaus Hansen. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

http://www.imavs.org/

Parameter Sensitivity in Control Allocation: A Monte Carlo Analysis for an Over-Actuated Aerial Robot

S. Al-zubaidi*, J. Skinner, and K. A. Stol
Faculty of Engineering, University of Auckland

http://www.imavs.org/

Abstract

Physical parameter uncertainty can degrade control allocation in over-actuated aerial vehicles by producing tracking error and cross-axis coupling. We present a Monte Carlo sensitivity study for a fixed-tilt octocopter, linking realistic motor/ESC, mass, inertia, centre-of-mass, and environmental variations to steady-state wrench error. For the tested platform and fixed allocator, vertical CoM displacement and inter-motor asymmetry dominate allocation degradation. The results provide practical guidance for mechanical tolerance control, actuator matching, and targeted calibration, while clarifying which conclusions are specific to this UAV and which may transfer to related over-actuated multirotor designs.

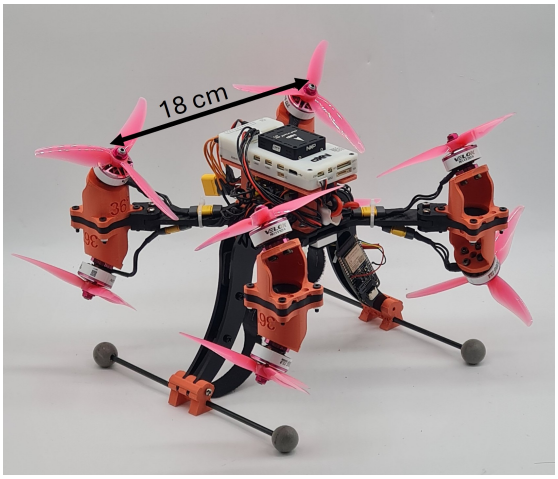


Figure 1: The stacked octocopter platform used in this study.

1 Introduction

Unmanned Aerial Vehicles (UAVs) have advanced significantly as demand for industrial applications grows [1]. As the scope of work for UAVs expands into precision agriculture, infrastructure inspection, and autonomous manipulation, demands for accuracy, safety, and performance have increased proportionally. The evolution of UAV airframe design has produced over-actuated configurations that enable control over all six degrees of freedom (DoF), allowing independent manipulation of position and orientation [2, 3].

Over-actuated UAVs, such as the example shown in Fig. 1, offer significant advantages over conventional quadrotors. The decoupling of position and orientation control enables precise end-effector positioning while maintaining arbitrary orientation, essential for applications such as aerial manipulation, inspection with directional sensors, or grasping tasks. Fully-actuated systems also provide higher bandwidth in horizontal axes that typically rely on attitude response in underactuated platforms, improving stability in dynamic environments such as turbulent wind [4]. Additionally, actuator redundancy provides fault tolerance and enables optimisation of secondary objectives such as power consumption or actuator wear.

However, over-actuated systems, in which the number of actuators exceeds the degrees of freedom, present challenges for control allocation. Because multiple actuator combinations can produce the same motion, the task of distributing control commands (known as *control allocation*) becomes more complex. Control allocation maps desired forces and moments from a high-level controller to actuator commands while respecting constraints such as saturation and rate limits [5]. For over-actuated systems where the effectiveness matrix $\mathbf{B}$ has more columns than rows, infinitely many solutions exist. Traditional methods, such as direct matrix inversion via the Moore-Penrose pseudoinverse, often perform poorly in these scenarios, yielding suboptimal solutions or, in critical cases, actuator saturation that degrades overall control performance [5].

Various approaches have been developed to address these challenges. Traditional methods include explicit ganging [6], daisy-chaining [7], and pseudoinverse-based mixing. The redistributed pseudoinverse iteratively clamps saturated actuators and recomputes the allocation, but early clipping errors can irreversibly degrade control performance [8]. Optimisation-based methods formulate control allocation as a constrained problem: direct allocation preserves commanded moment direction under saturation [9], while error minimisation directly prioritises tracking accuracy. Quadratic programming (QP) provides a convex, well-posed formulation that enforces actuator constraints robustly [10]. Adaptive methods address actuator uncertainty without explicit fault detection, with continuous adaptation modelling actuator ef-

*Email address: salim.al-zubaidi@auckland.ac.nz

http://www.imavs.org/

fectiveness as uncertain and updating allocation parameters using tracking errors [11]. Fault-triggered schemes integrate with fault detection systems to reconfigure allocation upon actuator failure [12].

For over-actuated UAVs, accurate knowledge of physical parameters (mass distribution, centre of gravity, inertia, and actuator characteristics) strongly influences control allocation performance. If the internal model is inaccurate, cross-axis coupling, tracking errors, and instability can result, particularly when using model-based control techniques such as Model Predictive Control (MPC) or Incremental Nonlinear Dynamic Inversion (INDI) [13]. These challenges are especially relevant for fully actuated and tilted-rotor multirotors, where non-vertical thrust components make allocation errors visible across multiple force and torque axes.

While allocation algorithms are well-studied, systematic quantification of parameter sensitivities remains limited. Most validation studies focus on conventional or underactuated platforms and assume known nominal parameters or focus on fault tolerance rather than the impact of realistic manufacturing tolerances and operational variations. Fixed-tilt multirotors and heterogeneous propulsion configurations introduce unique challenges including increased actuator coupling, nontrivial mapping between thrust inputs and resulting dynamics, and the potential for internal redundancy to obscure parameter observability [14]. Few works offer direct comparisons of allocation methods on common overactuated platforms or quantify which specific physical parameters most critically affect allocation performance, making it difficult to assess practical trade-offs in real deployment scenarios.

This paper addresses these gaps with a systematic, airframe-specific sensitivity analysis for a fixed-tilt stacked octocopter (Fig. 1). The main contribution is a Monte Carlo sensitivity analysis that identifies which physical parameters dominate allocation error, enabling targeted calibration, tolerance specification, and design optimisation. Parameter distributions are derived from manufacturer data, operational experience, and literature. A step-by-step simulation workflow connects sampled motor, ESC, airframe, CoM, and environmental parameters to steady-state force and torque output, clarifying how parameter deviations produce allocation errors. The resulting design guidance includes explicit bounds on generality: the numerical ranking is platform- and allocator-dependent, while the workflow can be repeated for other UAVs and allocation laws.

2 PLATFORM AND STEADY-STATE MODEL

2.1 Stacked Octocopter Platform

The platform is a custom-built over-actuated UAV with eight rotors arranged in two layers [14]. The upper rotors are tilted around the x-axis, while the lower rotors are tilted around the y-axis (Fig. 1). This fixed-tilt design provides full 6-DoF control authority without requiring variable-tilt mechanisms. The body frame follows the North-East-Down (NED) convention with the x-axis pointing forward, y-axis to the right, and z-axis downward.

Table 1: Nominal Physical Parameters

Parameter	Symbol	Value
Mass	m	0.95 kg
Inertia (Roll)	I_{xx}	0.024 kg·m^2
Inertia (Pitch)	I_{yy}	0.024 kg·m^2
Inertia (Yaw)	I_{zz}	0.105 kg·m^2
Arm Length	L	0.125 m
Rotor Tilt Angle	α	32°
Propeller Diameter	D_{prop}	5 × 4.5 inch

The hardware consists of BE-ox V32306 2250 kV motors with 5×4.5 inch propellers, 50A ESCs with bidirectional DShot. Table 1 summarises the key physical parameters.

2.2 Steady-State Wrench Model

In this study, control allocation is represented by a fixed nominal mixer matrix $\mathbf{\Gamma}$. The matrix is designed for the nominal geometry and is held constant during all uncertainty trials; therefore, the results quantify robustness of this control allocation method. The desired control setpoint $\tilde{\mathbf{u}}$ is mapped to individual motor commands as:

$$\mathbf{c}_{\text{norm}} = \text{clip}\left(\mathbf{\Gamma}\tilde{\mathbf{u}}, -1, 1\right) \tag{1}$$

The normalised command is scaled to the PWM input $\mathbf{u}_{\text{dyn}} = (\mathbf{c}_{\text{norm}} + 1)/2$. For each Monte Carlo trial, $\mathbf{\Gamma}$ remains unchanged while the physical plant parameters are perturbed. This captures a practical deployment scenario: an allocator is designed from nominal CAD and component data, but the real vehicle differs from that model.

The motor dynamic equation relates angular acceleration to electrical and mechanical torques:

$$J\dot{\omega} = \frac{K_t}{R}(uV_{\text{batt}} - K_e\omega) - K_t I_0 - C_\tau\omega^2 \tag{2}$$

where J is rotor inertia, K_t and K_e are torque and back-EMF constants, R is winding resistance, I_0 is no-load current, and C_τ is the propeller torque coefficient. At steady state ($\dot{\omega} = 0$), this quadratic balance is solved for the physically meaningful positive root $\omega_{ss,i}(u_i)$ for each motor. Thus K_t, K_e, R, I_0, C_τ, and ESC voltage mapping parameters influence the final wrench by first changing each motor's steady-state speed, then changing thrust and reaction torque through the squared-speed terms below.

Each motor produces thrust and reaction torque in its local frame:

$${}^{m_i}\mathbf{T}_i = \begin{bmatrix} 0 \\ 0 \\ -k_f\omega_{ss,i}^2 \end{bmatrix}, \quad {}^{m_i}\boldsymbol{\tau}_i = \begin{bmatrix} 0 \\ 0 \\ -d_{r,i}C_\tau\omega_{ss,i}^2 \end{bmatrix} \tag{3}$$

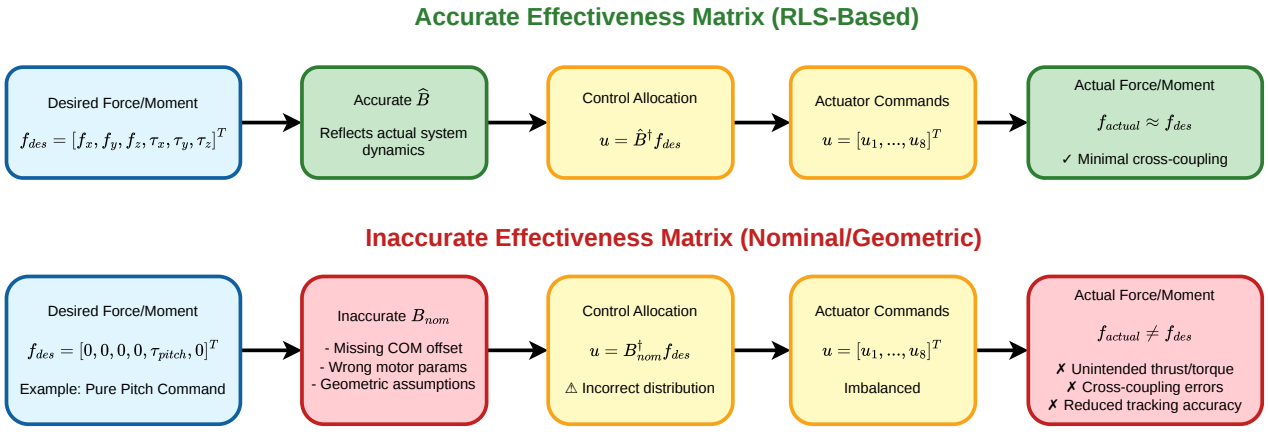


Figure 2: Control allocation error propagation: inaccuracies in the effectiveness matrix lead to incorrect actuator distribution, cross-axis coupling, and degraded tracking performance.

http://www.imavs.org/

where $d_{r,i} \in \{-1, +1\}$ indicates rotation direction and $k_f = 0.5\rho p_{z3} D_{prop}^2 A_p$.

The total body-frame wrench sums contributions from all motors, transformed from motor frames:

$$ {}^{b}\mathbf{T}_{ss} = \sum_{i=1}^{8} \mathbf{R}_{m_i \to b} \cdot {}^{m_i}\mathbf{T}_i \tag{4} $$

$$ {}^{b}\boldsymbol{\tau}_{ss} = \sum_{i=1}^{8} (\mathbf{R}_{m_i \to b} \cdot {}^{m_i}\boldsymbol{\tau}_i + \mathbf{r}_i \times (\mathbf{R}_{m_i \to b} \cdot {}^{m_i}\mathbf{T}_i)) \tag{5} $$

where $\mathbf{r}_i$ is the position vector from the Centre of Mass (CoM) to motor i. CoM perturbations enter through this moment arm, motor-axis geometry enters through $\mathbf{R}_{m_i \to b}$, and environmental and propeller parameters enter through k_f. Consequently, the same command vector can produce a different body wrench even though the allocator command is unchanged.

2.3 Simulation Workflow

Each simulation trial proceeds as follows:

1. Sample one vehicle instance from Table 2. Motor parameters are sampled independently; while the rest of the parameters are sampled at vehicle level.
2. Apply the fixed nominal mixer $\mathbf{\Gamma}$ to one representative setpoint to obtain clipped motor commands.
3. Convert each normalised command to a PWM input and solve Eq. 2 to find steady-state motor speed.
4. Convert each speed to local thrust and reaction torque, then transform these contributions into the body frame using Eqs. 4–5.
5. Normalise the resulting physical wrench and compute error and leakage scores.

3 Sensitivity Analysis Methodology

3.1 Sources of Parameter Uncertainty

Physical parameters are subject to variation from four primary sources:

- **Manufacturing tolerances**: Motor constants (K_V, K_E, C_τ) vary between units even within the same batch [15].
- **Assembly inconsistencies**: CoM shifts due to battery placement, cable routing, and component mounting.
- **Degradation**: Demagnetization reduces motor constants; bearing wear increases no-load current [16].
- **Environmental factors**: Air density varies with temperature, pressure, and altitude.

3.2 Monte Carlo Simulation

For each of $N = 10{,}000$ trials, every parameter was sampled from a normal distribution centred on its nominal value. The standard deviations in Table 2 are derived from manufacturer specifications, operational experience, and literature.

The 10% variation for I_0 (no-load current) is based on Portescap datasheet values and is assumed representative across brands.

Motor-specific parameters were sampled *independently for each motor* to capture inter-motor discrepancies. This enables analysis of both symmetric shifts (all motors deviate together) and asymmetric spread (motors differ from each other).

3.3 Justification of Selected Variations

The standard deviations in Table 2 model realistic operational uncertainty based on manufacturer data, literature, and operational experience.

Table 2: Parameter Variations for Monte Carlo Simulation

Parameter	Variation (σ)	Source
Motor & ESC (per-motor)		
K_V, K_E, C_τ	3.0%	[16]
R (Resistance)	5.0%	[15]
I_0 (No-load current)	10.0%	Portescap datasheet
V_{off}, m_v (ESC)	2.0%	[17]
D_{prop}	1.0%	[18]
Airframe & Mass		
Mass (M)	2.0%	[19]
Inertia (I)	5.0%	[20]
CoM X, Y	1 mm (abs)	[21]
CoM Z	10 mm (abs)	Operational
Environmental		
ρ_{air}	10.0%	Atmospheric

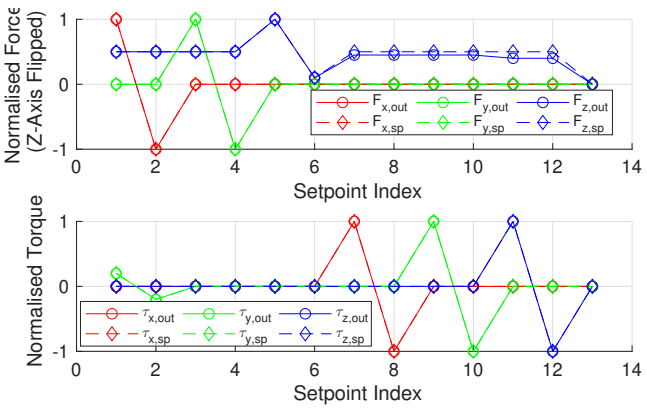


Figure 3: Normalisation of output wrench components. Physical forces (F_x, F_y, F_z in N) and torques (τ_x, τ_y, τ_z in Nm) are scaled to unitless ranges matching commanded setpoints: [-1, 1] for x/y forces and torques, [0, 1] for z-force.

Motor & ESC Parameters: Motor electromagnetic constants (K_V, K_E, C_τ) use 3% variation reflecting manufacturing tolerance and degradation (demagnetization) [16]. Resistance (R) uses 5% due to thermal effects and winding health variations [22]. No-load current (I_0) uses 10% due to bearing friction sensitivity. ESC voltage mapping (V_{off}, m_v) uses 2% from component tolerances [17]. Motor-specific parameters were sampled independently per motor to capture both symmetric shifts (all motors deviate together) and asymmetric spread (motors differ from each other). For each parameter in each trial, features were computed as: (1) *MeanDev*—mean absolute deviation from nominal across all motors, quantifying overall magnitude of shift; (2) *StdDev*—standard deviation of sampled values across motors, quantifying inter-motor variability; (3) *Range*—difference between maximum and minimum sampled values across motors, quantifying spread extremes. Symmetric shifts affect peak performance; asymmetric spread drives cross-axis leakage.

Airframe & Mass: Mass uses 2% for added hardware/cables beyond CAD. Inertia uses 5% from measurement and CAD estimation uncertainty. CoM uses absolute deviations $[1, 1, 10]$ mm (X,Y,Z), with larger vertical tolerance reflecting battery mounting variability (brand/size differences, velcro strap positioning).

Environmental: Air density uses 10%, combining barometric pressure fluctuations ($\pm$20–30 hPa $\rightarrow \pm 3\%$) and temperature swings (20°C $\rightarrow \pm 5\%$) [23].

3.4 Performance Metrics

Two metrics quantify allocation degradation. Since the desired setpoint $\mathbf{W}_{\text{desired}}$ is a unitless vector (normalised to [-1, 1] except for F_z which is [0, 1]), while the system output $\mathbf{W}_{\text{actual}}$ consists of physical forces (N) and torques (Nm), the output must first be normalised to enable meaningful comparison.

For the five axes with zero-centred command ranges ($F_x, F_y, \tau_x, \tau_y, \tau_z$), normalisation is performed by scaling against the vehicle's maximum authority:

$$[\mathbf{W}_{\text{actual, norm}}]_i = \frac{[\mathbf{W}_{\text{actual}}]_i}{[\mathbf{W}_{\text{max}}]_i} \quad \text{for } i \in \{1, 2, 4, 5, 6\} \tag{6}$$

For the vertical thrust axis (F_z), a min-max scaling accounts for a non-zero thrust offset identified during nominal characterisation:

$$[\mathbf{W}_{\text{actual, norm}}]_3 = \frac{[\mathbf{W}_{\text{actual}}]_3 - F_{z,\text{offset}}}{F_{z,\text{max}} - F_{z,\text{offset}}} \tag{7}$$

Figure 3 illustrates this normalisation, showing how physical force and torque outputs are mapped to the same unitless range as the commanded setpoints.

Normalised Error Score: The L2-norm of the difference between desired and normalised actual output:

$$S_{\text{Error}} = \|\mathbf{W}_{\text{actual, norm}} - \mathbf{W}_{\text{desired}}\|_2 \tag{8}$$

Relative Leakage Score: The magnitude of wrench components on axes commanded to zero, relative to the commanded wrench:

$$S_{\text{Leakage}} = \frac{\|\mathbf{W}_{\text{leakage}}\|_2}{\|\mathbf{W}_{\text{desired}}\|_2} \times 100\% \tag{9}$$

3.5 Test Setpoints

Six representative setpoints isolate each degree of freedom. All setpoints except pure vertical thrust include a 0.5 vertical thrust bias to operate around a typical hover point, linearising system behaviour around a common operating condition. Figure 3 shows the input setpoints.

http://www.imavs.org/

http://www.imavs.org/

3.6 Stepwise Multivariable Regression

To isolate each parameter's effect while accounting for confounding variables, stepwise multivariable linear regression with backward elimination was applied per setpoint [24].

For each setpoint, a full linear model predicts performance score (Y) from all parameter features ($X_1, \ldots, X_p$):

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon \tag{10}$$

Predictors with $p > 0.05$ were iteratively removed via backward elimination until all remaining coefficients were significant. Backward elimination is preferred over forward selection as it captures interactions that might appear insignificant in isolation. While LASSO regression offers advantages in some contexts, it can arbitrarily select among correlated predictors, which would complicate interpretation when motor parameters within the same unit are physically correlated.

Models achieved adjusted R^2 values ranging from 0.21 to 0.38 (mean 0.31). These modest values are an important limitation and indicate that a linear additive model does not capture the full nonlinear structure of the simulator. In particular, the features used here (MeanDev, StdDev, and Range) summarise the magnitude of motor mismatch but not which physical motor locations contain the high or low values; two samples with the same spread can produce different leakage depending on their spatial arrangement. We therefore interpret regression outputs as sensitivity indicators rather than predictive performance models. Future code releases will make it straightforward to replace this stage with higher-capacity models such as Gaussian processes, random forests, or polynomial chaos expansions, while retaining the same simulation workflow.

Final models were aggregated by *frequency* (percentage of setpoint models where a parameter remained significant) and *strength* (mean absolute t-statistic when significant), distinguishing localised vs. distributed effects.

4 Results

4.1 Nominal System Characterisation

The nominal system (all parameters at their specified values) was first characterised to establish baseline performance. A key finding was a non-zero thrust offset at zero command input, confirming linear response but requiring specialised normalisation for the F_z axis (Fig. 4).

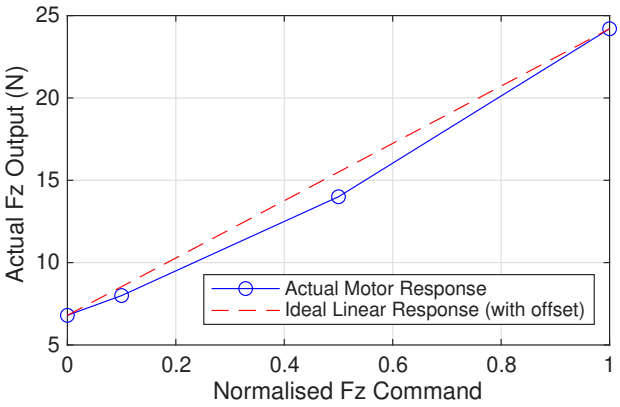


Figure 4: Physical thrust output vs. normalised F_z command for the nominal model.

4.2 Overall Parameter Sensitivity

Figure 5 aggregates results from all per-setpoint regression models, showing both the *frequency* at which each parameter was statistically significant and the *average strength* (mean absolute t-statistic) of its effect.

The results reveal a clear hierarchy of parameter sensitivities. Uncertainty in the vehicle's Centre of Mass (CoM) was found to be the dominant driver of both overall wrench error and cross-axis leakage. Secondary, yet still significant factors included inter-motor discrepancies, specifically the standard deviation of the back-EMF constant (`StdDev_KE`). Other motor parameters show smaller but consistent influences, with effects occurring uniformly across most manoeuvre setpoints rather than being manoeuvre-specific. In contrast, the system demonstrated robustness to environmental variations such as air density.

4.3 Key Findings

1. Vertical CoM Displacement Dominates: Centre of Mass variation along the Z-axis (`CoM_AbsDev_Z`) is the strongest performance driver for this platform. This parameter particularly affects horizontal thrust manoeuvres (F_x, F_y). When CoM shifts vertically, tilted thrust vectors no longer pass through the centre of mass, creating coupling between translation and rotation. For this platform's 32° tilt angle, a 10 mm CoM shift generates parasitic moments the allocator cannot fully compensate. This sensitivity is geometry-dependent and manoeuvre-specific.

Figure 6 illustrates this effect for the F_y setpoint (horizontal thrust), showing a strong linear relationship ($R = 0.89$) between vertical CoM deviation and roll torque leakage. The physical mechanism: vertical CoM displacement changes the moment arm for motors producing horizontal thrust, directly generating parasitic roll moments proportional to the displacement.

2. Motor Asymmetry Causes Consistent Leakage: Inter-motor variations, especially in back-EMF (`StdDev_KE`), represent the second most significant factor. Unlike CoM, motor asymmetry affects setpoints universally. When motors have different electromagnetic constants, identical commands produce unequal thrusts, creating unintended force/torque components. This affects fundamental actuator response rather than geometric coupling.

3. Distributed Motor Effects: Other motor parameters show smaller but consistent influences across both leakage and error metrics, with effects occurring uniformly across most manoeuvre setpoints rather than being manoeuvre-

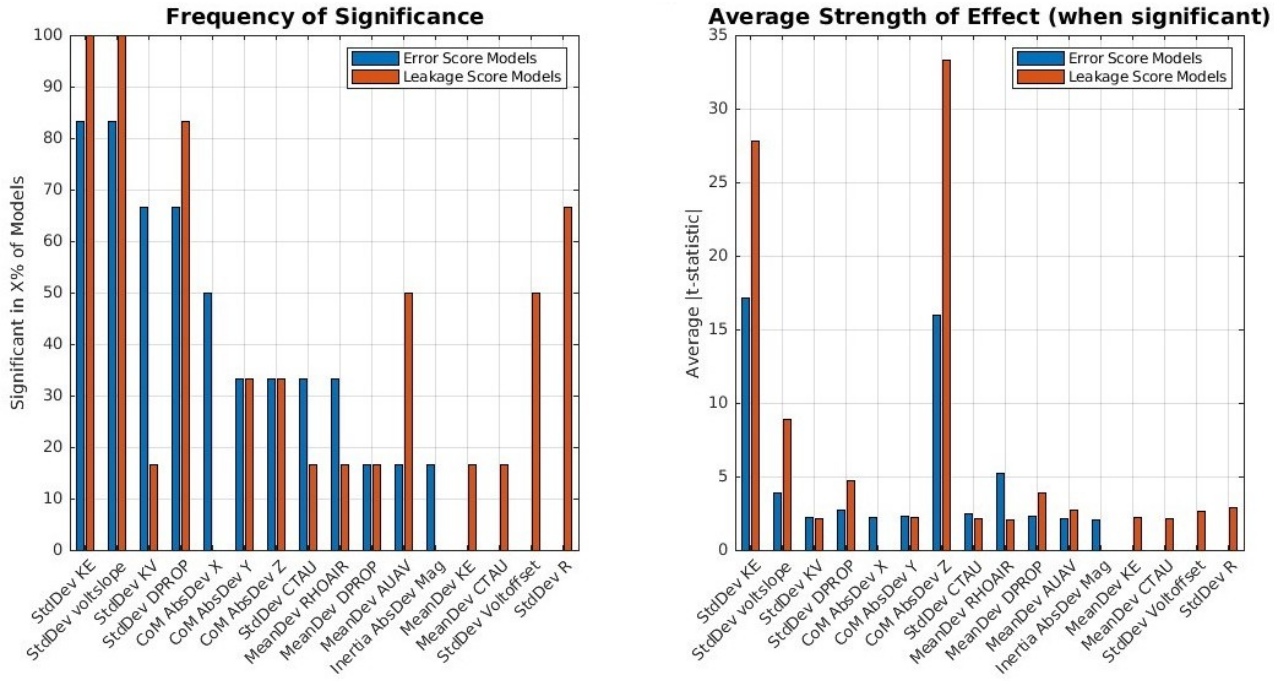


Figure 5: Overall parameter significance summary. **(Left)** Percentage of manoeuvre models in which each parameter remained significant after backward elimination. **(Right)** Average absolute t-statistic indicating typical effect strength.

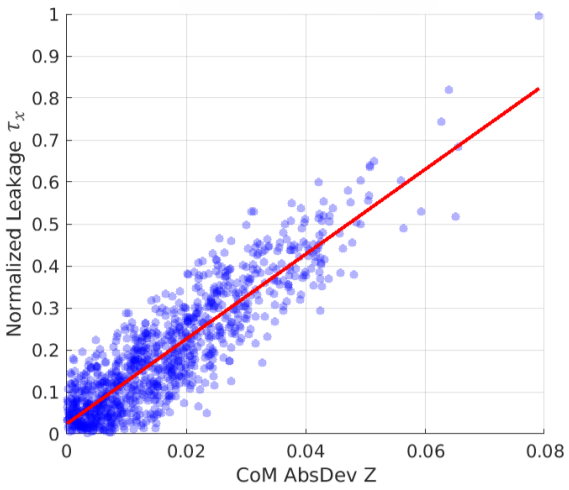


Figure 6: Vertical CoM deviation vs. normalised roll torque leakage for F_y setpoint (horizontal thrust).

specific.

4. Environmental Robustness: Air density was not statistically significant in any model. Symmetric changes (affecting all motors equally) preserve allocation geometry, so relative cross-coupling is unaffected unless approaching saturation.

5. Manoeuvre-Dependent vs. Universal Sensitivities: CoM affects specific manoeuvres (horizontal thrust), while motor asymmetry universally degrades performance. This distinction enables targeted mitigation: manoeuvre-dependent effects may be adaptively compensated, while universal effects require component matching or calibration.

5 Discussion

5.1 Implications for Practitioners

The results suggest the following prioritisation for improving control allocation accuracy in over-actuated systems:

Mechanical Precision - Highest Priority: First, standardising mass distribution to minimise CoM variation. Secondly, the use of mechanical fixtures (jigs, bays, locking mechanisms) rather than flexible mounting (straps, velcro) to ensure the parts do not move around during flight keeping CoM fixed during flight.

Component Matching: First, characterise individual actuator parameters and match units with similar characteristics. For electric motors, the back-EMF constant (K_E) is most critical, as asymmetry in this parameter drives consistent cross-axis leakage across all manoeuvres. In addition, distributing matched actuators balances asymmetric effects and reduces leakage errors.

5.2 Generalisability and Allocator Dependence

The specific sensitivity rankings depend on both this platform's geometry (32° tilt, stacked configuration) and a fixed matrix as the control allocation. However, the methodology

http://www.imavs.org/

http://www.imavs.org/

detailed in Section 3 can be repeated for other over-actuated UAVs and allocation methods. That said, several physical mechanisms are nevertheless expected to transfer across related aerial platforms:

1. CoM sensitivity: UAVs exhibit coupling between translational and rotational commands. This is particularly relevant for tilted-rotor and fully actuated multirotors. The dominant sensitivity axis varies with airframe geometry, but the underlying moment-arm mechanism is shared.

2. Actuator matching: Multi-rotor UAVs benefit from matched propulsion units, although the most critical motor or propeller parameter may vary with the airframe, motor model, and operating point.

3. Assembly vs. environment: Mechanical assembly precision is likely to dominate in controlled indoor UAV operation, while outdoor flight with large temperature, pressure, or wind variations may increase the relative importance of environmental uncertainty.

The Monte Carlo workflow presented here provides a template for quantifying sensitivities on other UAV platforms. The regression stage gives a compact, statistically interpretable summary, despite strong correlations. For this platform, the strongest mechanism was also visible directly in the simulator output, as evidenced by the strong correlation between vertical CoM deviation and observed roll leakage.

5.3 Experimental Check and Limitations

This analysis focuses on steady-state allocation performance and does not capture transient effects from motor dynamics (time constants, delays), aerodynamic rotor-rotor interactions or rate-constrained allocation behaviour near saturation. The regression analysis assumes linear superposition of parameter effects; nonlinear interactions may exist at extreme parameter deviations. The allocator is also fixed throughout the study, so conclusions about relative sensitivity should be re-evaluated for allocation methods with different constraint handling, null-space objectives, or adaptation.

A limited qualitative experimental check was conducted on a similar physical octocopter using motion-capture measurements during horizontal thrust manoeuvres with intentionally shifted battery/CoM placement. The observed cross-coupling direction was consistent with the simulation prediction that vertical CoM displacement induces parasitic roll/pitch moments during lateral force commands. This is not presented as quantitative validation because closed-loop controller compensation, motion-capture differentiation noise, and imperfect knowledge of the exact CoM shift prevented a clean comparison.

6 CONCLUSION

This paper presented a systematic uncertainty-to-wrench sensitivity workflow for quantifying how realistic parameter variations affect a fixed control allocator on an over-actuated fixed-tilt octocopter. Using Monte Carlo simulation with 10,000 samples and interpretable regression screening, we identified vertical Centre of Mass displacement and inter-motor asymmetries as the dominant factors degrading allocation accuracy and inducing cross-axis leakage for this platform and allocator.

For the tested platform, physical assembly precision and component matching had stronger effects on control allocation performance than aggregate mass properties or ambient conditions. Tightening vertical CoM tolerance (e.g., through standardised battery mounting) and matching motor electromagnetic constants offer practical routes to improved allocation accuracy.

These findings translate into design guidance for over-actuated UAVs: prioritise mechanical consistency in mass distribution, characterise and match actuators before assembly, and focus calibration on inter-unit variations rather than environmental compensation. While the specific sensitivity rankings depend on airframe geometry and allocation law, the workflow can be applied to related multirotor UAV designs.

Future work will extend this analysis in four directions. First, incorporating motor dynamics and transient responses will capture time-dependent effects during rapid manoeuvres. Second, load-cell or thrust-stand validation will quantify the predicted sensitivity rankings under controlled parameter changes. Third, repeating the workflow for pseudoinverse, constrained QP, redistributed, and adaptive allocators will show how allocation design could change uncertainty margins.

ACKNOWLEDGMENTS

The research 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] T. P. Nascimento and M. Saska. Position and attitude control of multi-rotor aerial vehicles: A survey. *Annual Reviews in Control*, 48:129–146, 2019.

[2] P. H. M. Souza and K. Stol. Constrained dynamics of an aerial manipulator interacting with flexible cantilever beams. *IEEE/ASME Transactions on Mechatronics*, 28(2):967–975, 2023.

[3] R. Rashad, J. Goerres, R. Aarts, J. B. C. Engelen, and S. Stramigioli. Fully actuated multirotor UAVs: A literature review. *IEEE Robotics & Automation Magazine*, 27(3):97–107, 2020.

[4] Z. J. Chen, J. X. J. Bannwarth, K. A. Stol, and P. J. Richards. Analysis of a Multirotor UAV with Tilted-Rotors for the Purposes of Disturbance Rejection. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 864–873, Dallas, TX, June 2018. IEEE.

[5] M. Bodson. Evaluation of optimization methods for control allocation. *Journal of Guidance, Control, and Dynamics*, 25(4):703–711, 2002.

[6] M. W. Oppenheimer, D. B. Doman, and M. A. Bolender. Control allocation for over-actuated systems. In *2006 14th Mediterranean Conference on Control and Automation*, pages 1–6, 2006.

[7] J. M. Buffington and D. F. Enns. Lyapunov stability analysis of daisy chain control allocation. *Journal of Guidance, Control, and Dynamics*, 19(6):1226–1230, 1996.

[8] J.D. Boskovic and R.K. Mehra. Control allocation in overactuated aircraft under position and rate limiting. In *Proceedings of the 2002 American Control Conference*, volume 1, pages 791–796, 2002.

[9] W. C. Durham. Attainable moments for the constrained control allocation problem. *Journal of Guidance, Control, and Dynamics*, 17(6):1371–1373, 1994.

[10] M.F. Santos, L.M. Honório, A.P.G.M. Moreira, P.A.N. Garcia, M.F. Silva, and V.F. Vidal. Analysis of a fast control allocation approach for nonlinear over-actuated systems. *ISA Transactions*, 126:545–561, 2022.

[11] S. S. Tohidi, Y. Yildiz, and I. Kolmanovsky. Adaptive control allocation for over-actuated systems with actuator saturation. *IFAC-PapersOnLine*, 50(1):5492–5497, 2017.

[12] J. B. Davidson, F. J. Lallman, and W. T. Bundick. Real-time adaptive control allocation applied to a high performance aircraft. In *Society for Industrial and Applied Mathematics*, United States, 2001.

[13] E. J. J. Smeur, Q. P. Chu, and G. C. H. E. De Croon. Adaptive incremental nonlinear dynamic inversion for attitude control of micro air vehicles. *Journal of Guidance, Control, and Dynamics*, 39(3):450–461, 2016.

[14] 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.

[15] A. I. Abouseda, R. O. Doruk, and A. Amini. Parameter identification and speed control of a small-scale BLDC motor: Experimental validation and real-time PI control with low-pass filtering. *Machines*, 13(8):656, 2025.

[16] D. K. Athanasopoulos, P. D. Karagkounis, J. C. Kappatou, and S. Tsotoulidis. Demagnetization faults analysis in a BLDC motor for diagnostic purposes. In *2014 International Conference on Electrical Machines (ICEM)*, pages 1862–1868, 2014.

[17] A. Gong, R. MacNeill, and D. Verstraete. Experimental testing of electronic speed controllers for UAVs. In *2017 IEEE International Conference on Unmanned Systems (ICUS)*, pages 88–93, 2017.

[18] M. Etewa, A. F. Hassan, E. Safwat, M. A. H. Abozied, M. M. El-Khatib, and A. Ramirez-Serrano. Performance estimation of fixed-wing UAV propulsion systems. *Drones*, 8(9):424, 2024.

[19] R. Sanz, P. Garcia, Q.-C. Zhong, and P. Albertos. Robust control of quadrotors based on an uncertainty and disturbance estimator. *Journal of Dynamic Systems, Measurement, and Control*, 138(7):071006, 2016.

[20] T. Bresciani. Modelling, identification and control of a quadrotor helicopter. *Master's Thesis, Lund University*, 2008.

[21] X. Yu, Y. Zhang, and J. Jiang. Experimental study on adaptive control for quadrotors with unknown payload and center of mass offset. *Control Engineering Practice*, 136:105555, 2023.

[22] T. A. Shifat and H. Jang-Wook. Remaining useful life estimation of BLDC motor considering voltage degradation and attention-based neural network. *IEEE Access*, 8:168414–168428, 2020.

[23] J. D. Anderson. *Introduction to Flight*. McGraw-Hill Education, 8th edition, 2016.

[24] N. R. Draper and H. Smith. *Applied Regression Analysis*. Wiley, 3rd edition, 1998.

http://www.imavs.org/

A Multi-View Multi-LiDAR Synchronised Setup for Smoke Plume Data Acquisition: A Simulated Drone Swarm Observation Case Study

G. Pérez*[1], L. Denis[1], H. Chakraa[2], M. Bronz[2]

[1]LAAS-CNRS, Université de Toulouse, Toulouse, France.
[2]École Nationale de l'Aviation Civile (ENAC), Université de Toulouse, 31400 Toulouse, France.

http://www.imavs.org/

ABSTRACT

Each year, hundreds of thousands of hectares are devastated by wildfires, causing severe impacts on biodiversity and major economic losses. In this context, better understanding of wildfires for rapid and effective response is essential. We aim to develop methods of wildfire analysis using a drone fleet to provide accurate information for optimal intervention. This work forms part of this project with a focus on smoke analysis. The chaotic behaviour of fires, and the high temperatures makes the safe deployment of drones in a wildfire context a challenging task, leading to a lack of data for smoke and fire analysis. To address this gap, we present an experimental setup leading a public dataset of temporally synchronised and spatially calibrated images and point clouds of smoke plumes in a controlled environment. Using three 3D LiDARs Livox-MID360 and five heterogeneous HD cameras, this setup is the first to make multi-view and multi-LiDAR data of smoke scenes available[1]. A proof-of-concept demonstration is presented, featuring a simulation of a swarm of drones interacting with a dynamic smoke plume represented by a 3D point cloud obtained with the described setup.

1 INTRODUCTION

1.1 Context

Due to climate change and rising summer temperatures, which are breaking new records every year, wildfires are becoming increasingly frequent. Whether caused by arson, accident or natural causes, they are extremely devastating and seem difficult to predict. These extreme fires have a direct impact on the natural environment and the health of living beings (including humans), as they are very difficult to control and release toxic fumes that severely affect air quality [1]. Given the increasing frequency of wildfires, it is essential to be prepared and to manage these fires as effectively as possible in order to minimise their socio-ecological impact [2]. In this context, a drone fleet equipped with LiDARs [2] and cameras could gather data of the smoke to analyse it and try to model in order to predict the short-term evolution of the smoke plume and the fire. This information could prove useful to firefighters on the ground in helping them devise a strategy to contain the fire. To our knowledge, we are the first to build a dataset combining LiDAR and images to capture smoke rising, allowing to explore diverse methods for smoke understanding. This opens the door to exploit new kind of data for drone control or machine learning tasks.

1.2 Related work

The presented setup tries to link computer graphics/computer vision works on smoke reconstruction, to potential robotics applications that can be done with drones in the context of wildfires.

LiDAR for environment observation. Since several years now, LiDARs have become a central component in several robotics fields and for different purposes. In autonomous vehicles fields, combining several LiDARs enables to build a precise map of the surrounding environment with minimised blind spots, making localisation, object tracking and decision-making more robust [3] [4] [5]. LiDARs were also impressively impactful in agriculture fields since it can help to monitor and characterise the state of crops and trees [6], using different types of vehicles [7][8][9][10].
As LiDARs are mainly used in robotics for object detection and navigation, data accuracy is critical, in this context some work show interest to the interaction with aerosols or in adverse weather, but usually considering it as noise that reduces performances [11] [12] and not to observe their properties.

Camera-LiDAR sensor fusion. Even if 3D LiDAR is now a widely used component in robotics they have, like all sensors, limitations in some cases. The discrete representation of the environment and the absence of colour information makes it hard to characterise the targets. It is then widely recognised that sensor fusion gives more promising information. Specifically in the case of LiDARs, Camera-LiDAR sensor fusion was the first association to rise in literature starting from the

*Email address: gaetan.perez@laas.fr
[1]Git repo available here : https://github.com/Typos91/LiCamFlow

[2]Please note that when speaking about LiDARs or 3D LiDARs, we refer to mid-range sensors (around 100m), unless otherwise stated.

pioneer KITTI dataset [13].
Even though this sensor combination is majorly used in the case of autonomous vehicles, and object detection, we found very few works referring to Camera-LiDAR sensor fusion in case of dynamic phenomenon observation, and more specifically fluid flows like in this study.

Fluid flow Observation. Fluid flow observation is an important concept in various fields like fluid mechanicals but also computer graphics or machine-vision. Several techniques to extract the volumetric fluid flow information were designed in the past. One of the most famous techniques was the Particle Imaging Velocimetry (PIV), it regroups a set of techniques reviewed first by [14] which uses lasers illuminating a fluid flow with tracers particles, and capturing the scattering effect of the laser on these tracers across time, to get an information of the local dynamics of the fluid and following it in time. A similar approach was to capture the scattering effect of a flow particles (without tracers) in participating media like smoke [15] [16]. However, the scattering effect depends on the relative size of the particles in relation with the laser wavelength making this technique dependent of the composition of the media studied. PIV technique is a pioneer in fluid flow volumetric observations as it enables to perform 3D reconstruction of the flow using multiple cameras [17]. However, the observation is usually made on small volumes and needs an important setup that must be installed in a laboratory which is therefore not adapted in our case.
A widely used alternative to these setups involving lasers, and usually made for small-scale studies, is the tomography. It consists of sets of 2D views of the subject, which are used to build a 3D volumetric reconstruction. Tomography proved to be very useful especially for gas and smoke observation, which is more complex to observe with the PIV method. [18] uses optical flow algorithm and a specific noisy background to record transparent non-stationary gas flow with 16 cameras, and builds a 3D reconstruction of refractive index of the gas enabling to determine physical properties of the gas. [19] takes low-resolution 2D views as an input to flow tracking by adding physical constraints through the Navier-Stokes equation and a pressure solver usually used in simulations. Recently, Saclarflow [20] became a huge milestone specifically in 3D smoke reconstruction, presenting physically-based simulated smoke voxel grids trained with real data. By releasing the dataset they used to train their model, it opened the door to new machine learning-based tomography techniques [21] for smoke flow observation. This work aims to follow on from fluid flow tomography works, and add LiDAR data, giving geometrical precision to the observations. We think that these data may be of interest outside the computer graphics community with the addition of LiDAR data. At larger scales, other types of sensors can be used for flow observation. More specifically with smoke, RADARs and Doppler LiDARs are used to observe smoke density and speed information at long range [22], enabling to study precisely the speed fields inside a wildfire smoke plume. However, these sensors are not made for mobile robotics and need to be static to be used.
Other methods involving images for smoke plume observation can be used, such as satellite imagery or using a fleet of drones. [23] deploys a fleet of drones, in which a leader drone observes a plume using a camera and controls four other drones to optimise observation. These drones use only cameras to observe the plume and then generate a 3D coloured point cloud from the images recorded using the COLMAP algorithm. However, our work and objectives focus on the use of a camera-LiDAR combination to directly get a 3D point cloud from the sensors and improve the quality of the generated point cloud obtained when using solely COLMAP.

Datasets. In both robotics and computer graphics, datasets are of great importance since they enable testing algorithms developed with real-world conditions, showing the limits but also opening new subjects. However in robotics, those algorithms are generally not used directly for control but for closely related topics like computer vision tasks or localisation.
As mentioned previously, the Scalarflow [20] dataset had major a impact on the computer graphics community in smoke observation; however, the lack of situation diversity in the sequences prevents proper generalisation.

2 Multi-view and multi-LiDAR acquisition

In order to build a coherent dataset, we want to be able to merge image and point cloud data. These can be used if sensors are spatially calibrated (meaning that the frame transformation between each sensor is known) and temporally synchronised.

2.1 Setup

The complete setup consists of 3 "acquisition modules" plus 2 cameras placed all around a fog machine to minimise blind spots. An acquisition module corresponds to a fixed piece with 1 LiDAR Livox MID-360 and 1 camera attached. The two remaining cameras are not attached to any LiDAR. For the LiDARs to be able to capture the whole plume, they are rotated 90° so that the horizontal view becomes the vertical one, and therefore the vertical FOV is no longer a limitation.

Choice of LiDAR. The choice of the LiDAR is crucial, as it is a balance between what can be assembled on a drone and the quality of the output data. For this choice, 2 LiDARs were compared with respect to their mechanical properties (size of sensor and weight) and the data quality: the Livox MID-360 and the Ouster OS1. Figure 2 gives a visual of the comparison between the two sensors. Two launches of smoke were performed using the fog machine, one with a Livox MID-360 LiDAR's recording (2a 2b 2c), and the other using an Ouster OS1 LiDAR (2d 2e 2f). In figures 2e 2f and 2b 2c, the plume was extracted from the rest of the point cloud to focus the comparison on it. The smoke extraction is explained in section 3.2.

http://www.imavs.org/

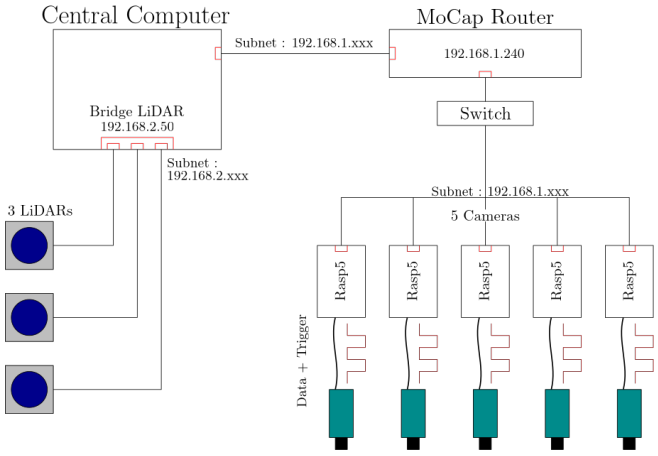
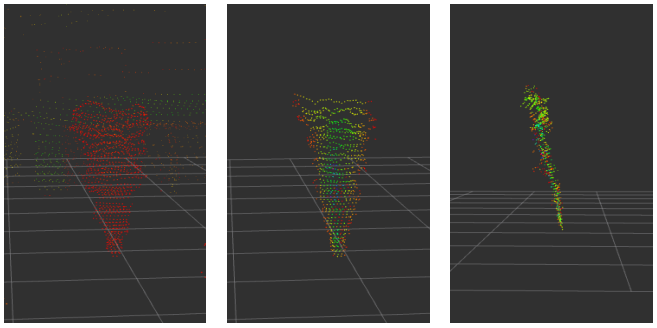
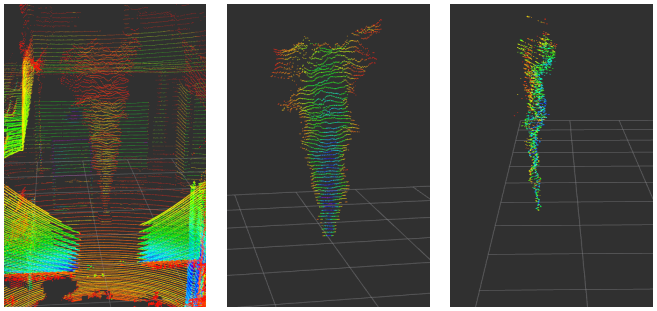


Figure 1: Schematic diagram of the final setup for multi-view/multi-LiDAR acquisition



(a) Livox plume, full point cloud (b) Livox plume, front view (c) Livox plume, side view



(d) Ouster plume, full point cloud (e) Ouster plume, front view (f) Ouster plume, side view



(g) Rainbow colorbar from low to high intensity/reflectivity

Figure 2: Comparison of a smoke plume detected with 3 Livox MID-360 LiDARs (up) and a smoke plume detected with one Ouster OS1 LiDAR (bottom). The colour bar represents the evolution of the point colours with respect to their intensity (or reflectivity). This information is provided directly by the sensor, however, each sensor can have a different way to compute this value, so they cannot be compared directly.

Unlike the OS1 sensors, the MID-360 LiDAR employs a different, non-repeatable scanning method, leading to some holes in the 3D point cloud of the smoke plume, as it can be seen in figure 2a. However this pattern allows for more random distribution of points within the environment, providing an accurate representation with fewer scans (a maximum of 200,000 points per second for the Livox MID-360 compared to a maximum of 5.8 million points per second for the OS1). This difference in point number is clear when comparing figure 2a with 2d. This is also reflected in the number of points representing the smoke plume, which is around 2000 points for the Ouster LiDAR, whereas the Livox returns around 1000 points for the same smoke plume. It is also worth noting that the reflectivity given by the Livox LiDAR is harder to use because all points have very low value. On a scale from 0 to 255, with 0-150 values corresponding to diffuse scattering objects and 151-255 corresponding to total reflection objects, the returns of the smoke have their intensity value usually between only 0 and 10 (please note that the computation of the reflectivity varies depending on the LiDAR manufacturer, but it is always computed using the ratio of the amount of energy received relative to that emitted by the sensor).
However, the strength of the Livox LiDAR resides in its size ($65 \times 65 \times 60mm$) and its small weight ($285\ g$ with integrated mounting system versus $925\ g$ with mounting system for the Ouster OS1), making it very appropriate for small drones easily deployable, which is an important characteristic in our context. Therefore, even if the Livox MID-360 shows some limitations in comparison with the OS1, its small size and adaptability to drone applications make it a better choice for our setup.

Network. This setup aims to be adaptable to different conditions, in particular, indoor and outdoor conditions in order to compare data in controlled and uncontrolled environments. Outdoor conditions also imply longer distances, which means longer cables for data transmission. Therefore, an Ethernet connection must be used. Since USB3.0 cameras are used, a Raspberry Pi 5 is added to each camera. The cameras are directly connected to their Raspberry Pi, and the latter are connected via Ethernet to a switch connected to a router. Furthermore, the Raspberry Pi will be used for temporal synchronisation using the GPIOs to trigger the camera. Figure 1 presents a schematic diagram of the components and the networking done for this setup. As shown in the figure, all elements (except for the LiDARs) are connected to a router, which assigns to them an IP address. This subnet is then used to access the Raspberry Pis using an SSH connection from the main computer.

Lighting. The dataset targets image-LiDAR connection, and handling semi-transparent interactions with the back-

http://www.imavs.org/

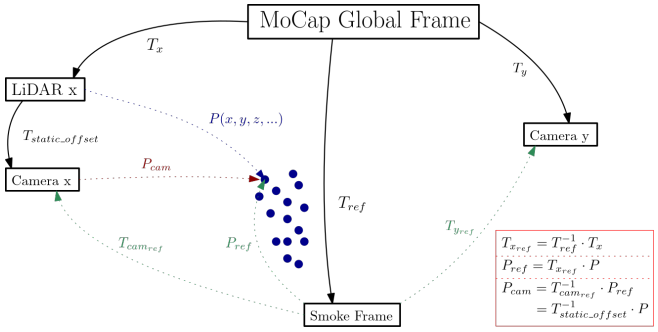


Figure 3: Scheme of the transformations and known positions given by MoCap. The black lines refer to the position given by the MoCap. The blue lines refer to positions in the LiDAR frame. The green lines are the position given in the reference frame, here the fog machine. The red lines refer to the position given in the camera frame from where the line starts. Transformations T are 4×4 homogeneous matrices containing translation and rotation between two frames. And Ps are the coordinates of a point given by the LiDAR, in a given frame.

ground is out of scope. For this reason, we focus on images where the light makes our understanding easier, and the background must have as little interaction as possible. This approach is inspired by Scalarflow [20] but differentiates by the size of the plume (ours elevates at five metres high, compared to approximately one metre in the Scalarflow dataset). As the sensors are disposed in a circle all around the smoke in order to capture not only one face of the smoke but also the whole plume, a black cloth cannot be set easily, and therefore the background is visible. However, to tackle this problem, the lights of the room are turned off, and 4 LED spots are posted directly towards the smoke plume in such a way that they do not appear behind the smoke for any camera. With this configuration, we obtain a nice contrast between the useless background and the subject of the images, which will be useful for the post-processing part.

2.2 *Sensor calibration*

Many pipelines in computer vision require precise camera calibration and knowledge of extrinsic and intrinsic parameters. Intrinsic parameters refers to internal characteristics of the sensor. In the case of a camera the parameters are the focal lengths, the central point of the sensor and the distortion effect of the lens. As for the extrinsic parameters, they refer to the position of the sensor in a given frame (translation and orientation). Therefore, the sensors were carefully calibrated using classical chessboard calibration for the intrinsic parameters of the cameras. As for the extrinsic parameters (cameras and LiDARs), motion capture was used.

Motion Capture. When working with cameras and LiDARs, especially when the objective is to use them for data fusion, it is important to have a precise estimation of the sensors positions, within the same frame. Using a chessboard detectable by LiDARs and the cameras, it is possible to find every position relative to one arbitrarily chosen sensor. However, this technique is heavy to set up and must be redone each time a sensor is moved intentionally or accidentally. When doing a setup with dozens of sequences, a dynamic way to have the sensors positions is needed. The motion capture is perfectly suited for this task, as it is able to give real-time positions of the sensors using markers like those presented in figure 4. The alignment between the motion capture rigid body origin and the sensors internal frame was done using the dimensions of the sensors given in their data-sheet meaning that a static offset between the motion capture rigid body and the actual sensor frame exists. Data is provided to do a more precise calibration using a reflective chessboard with motion capture markers.

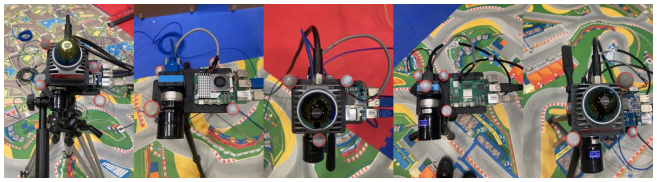


Figure 4: Motion Capture markers (circled in red) on each module

2.3 *Temporal synchronisation*

To observe a physical phenomenon, it is necessary to observe the evolution of the system over time. If several sensors are used, they must be synchronised so that the data can be merged with certainty that the same phenomenon is being detected by each sensor. This is particularly important for a plume of smoke, which is highly sensitive to the slightest disturbance.

Since perfect synchronisation is impossible, we estimate an offset acceptance. Using the first LiDAR frames of a sequence, when the smoke is expelled from the fog machine and rises, we measure the approximate speed of the rising smoke thanks to the known timestamps of the point clouds, which will give an approximation of the smoke speed at the source. With this technique, the estimated smoke speed is approximately $v_{smoke} = 3\text{m/s}$. According to [24], which carried out a comprehensive study that verified the sensor specification, the Livox MID-360 has centimetre accuracy. Therefore, with this precision limitation we can compute a temporal offset acceptance: $\delta t = \delta x / v_{smoke}$. With the LiDAR precision $\delta x \approx 0.01m$.

Therefore, we obtain a maximum wanted time offset of $\delta t \approx 3.3ms$ between sensors.

Camera triggering. In order to synchronise the shots, [20] uses the same hardware trigger for all cameras. In our case, as we want a setup that can be adapted to long distances, a single cable for all the cameras seems more complicated to manage.

http://www.imavs.org/

We choose to parallelise the trigger signals, meaning that each Raspberry Pi associated with a camera will be responsible for sending the signal. In order for all signals to be sent at the same time, all Raspberry Pi units must be synchronised (i.e., hey must have the same time reference) and receive an instruction indicating when the signal should be sent. All this must be done while keeping in mind that the shots must also be synchronised with the LiDAR scans. The latter operate at 10Hz and have a scan time of just under 0.1s. We want to be able to take 2 photos for each scan, which means a frequency of 20Hz for the cameras. Based on these constraints, we construct the algorithm 1. We use the actual difference between two LiDAR scans to estimate when to send a signal to trigger the cameras. Algorithm 2 describes how to obtain the average value of the timestamps of the three LiDARs and compute the mean difference between two scans of a LiDAR.
For this algorithm to work properly, all devices must be synchronised so that they all have the same time reference. A central clock is chosen, to which all other devices (Raspberry Pi and LiDAR) can synchronise.

Synchronisation protocol. A synchronisation protocol is a protocol used to synchronise a local clock with that of a server connected to a physical clock (usually an atomic clock). For example, the NTP (Network Time Protocol) protocol allows synchronisation with a server that is accessible worldwide. These synchronisation protocols generally work with UDP communication protocols, hence the need for an Ethernet connection. For synchronisation with reference to a local physical clock, the PTP (Precision Time Protocol) synchronisation protocol is used [25].

2.4 Acquisition

Image Writing. One of the challenges when storing data received by a Raspberry Pi is its write speed. Indeed, the data must be stored faster than the rate at which it is received. This depends on several factors, such as the image size, its encoding, and the frame rate. In our case we work with images measuring a maximum of 1920 × 1200 px, with GRAY8 encoding (greyscale, 8-bit) and at a frame rate of 20 frames per second, which corresponds to a writing speed of 46 MB/s. This speed is below most of the SD card's maximum capacity, but if the frame rate were to be increased, a more specific storage system should be used.

ROS2 integration. ROS2 is a middleware platform used for the development of robotic systems. It enables the exchange of information between multiple programmes, known as nodes, via various standardised methods (topics, services and actions). Thus, for multi-sensory experimentation, it is a very adapted tool, especially when the various sensors are connected in some way (for example, with synchronisation or spatial calibration). Furthermore, ROS2 offers visualisation tools such as Rviz2 or RQT, which facilitate a comprehensive understanding of the system.
Therefore, ROS2 was used to record the LiDARs as well as

Algorithm 1: LiDARs - Cameras synchronisation using PTP and Raspberry Pi

Inputs :
$\Delta T_{\text{camera}_j}$: Time Exposure of the camera i
N : number of LiDAR
K : number of cameras
Temporal Variables:
$T_{\text{ts\_LiDAR}_i}$: LiDAR timestamp i
T_{capt} : Mean Timestamp between each LiDAR's timestamp
ΔT_{LiDAR} : Mean duration between 2 scans
t : Current Time of the computer
Outputs : Trigger signal sent from each raspberry Pi to their corresponding camera

Initialisation :
Start PTP protocol from the computer (master) to synchronise the devices clock (LiDARs and Raspberry Pi)
Wait until time is synchronised (indicated in LiDAR message)
For each LiDAR read $T_{\text{ts\_LiDAR}_i}$
for $t \leftarrow t_0$ **to** $t_0 + 1s$ **do**
  $T_{\text{capt}}, \Delta T_{\text{LiDAR}} \leftarrow$ `get_mean_timestamp(`t, N`)`
end
Starting Acquisition :
while *Aquisition active* **do**
  for $j \leftarrow 1$ **to** K **do**
    Compute $T_{\text{trigger}_j} \leftarrow T_{\text{capt}} - \frac{\Delta T_{\text{camera}_j}}{2} + \Delta T_{\text{LiDAR}}$
    `// Estimate next time to Trigger each camera corresponding to its time of exposure`
  end
  $T_{\text{capt}}, \Delta T_{\text{LiDAR}} \leftarrow$ `get_mean_timestamp(`t, N`)`
  for $j \leftarrow 1$ **to** K **do**
    while `PTP_synced_clock()` $< T_{trigger_j}$ **do**
      Wait
    end
    Send Trigger `// This must be done in parallel thanks to the PTP calibration between the Raspberries`
  end
end

http://www.imavs.org/

Algorithm 2: get_mean_timestamp

Inputs : t : Current Time of the computer
N : number of LiDAR
Outputs: T_{capt} : Mean Timestamps between all LiDARs
ΔT_{LiDAR} : Mean duration between 2 scans

for $i \leftarrow 1$ **to** N **do**
  Read $T_{\text{ts\_LiDAR}_i}$ from ROS2 message of LiDAR i
  `// recall that LiDARs and computer's clocks are synchronised`
end
Compute $T_{\text{capt}} \leftarrow mean(T_{\text{ts\_LiDAR}_i})$ `// Compute mean of timestamps`
Compute $\Delta T_{\text{LiDAR}} = T_{\text{capt}}(t) - T_{\text{capt}}(t - \Delta T_{\text{LiDAR}})$
Return :
$T_{\text{capt}}, \Delta T_{\text{LiDAR}}$

the motion capture positions, enabling the computation of the position of each sensor in the same frame with the smoke centred in (0,0). It was also used to send the information of when to trigger the cameras to each Raspberry Pi.

3 DATA POST PROCESSING

Different types of setups were tested, all respecting the imposed constraints (sensors all around the smoke, temporal synchronisation, spatial calibration, ...). So, the first step of the post-processing step was to qualitatively evaluate the data in order to strike a balance between data quality and ease with which the experimental setup can be reassembled in the same or different conditions.

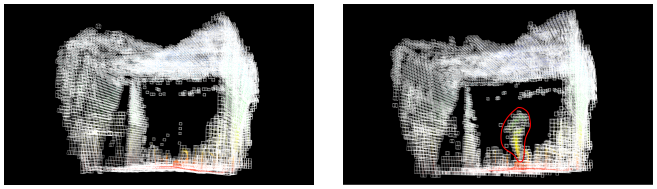
3.1 Images

As smoke is a semi-transparent substance, it is difficult to isolate, especially when the background is not plain. Therefore, we apply simple filters to try to remove the background without degrading too much the quality of the smoke in the image. The first step is to compute the median value of the background using images without smoke and then to subtract this from the images containing smoke. We finally apply a threshold filter to complete the background de-noising.

3.2 Point clouds

Merging clouds. To visualise the smoke as a single point cloud, the three point clouds sent by the LiDARs must be merged. To do so, the position given by the motion capture system is used to project all the points in the same global frame. Figure 3 gives the transformation matrices to apply to project a point on the global frame ($ref\ frame$ in the figure). Another possibility which was not used here is to use the ICP algorithm with all LiDARs 2 by 2 to find all the needed transformations.

Labelling and Smoke extraction. The first challenge in processing LiDAR data once all point clouds are merged is the segmentation. More specifically, our aim is to identify which points correspond to the smoke plume so that we can extract them from the rest, as shown in figure 2, or just label them.



(a) Octree reference map without the smoke

(b) Octree map with new smoke voxels

Figure 5: Visualisation of the octree map created to label and extract the points corresponding to the smoke plume.

There are several segmentation algorithms, such as those based on the RANSAC algorithm, which attempts to identify specific shapes within the point cloud [26]. Other machine learning-based methods improve segmentation performance by also detecting dynamic objects [27]. In our case, we are seeking to extract a dynamic object with a potentially highly variable shape, rendering RANSAC-based algorithms ineffective. Furthermore, given the limited (or even non-existent) availability of LiDAR smoke data, the use of machine learning-based algorithms appears ill-suited.

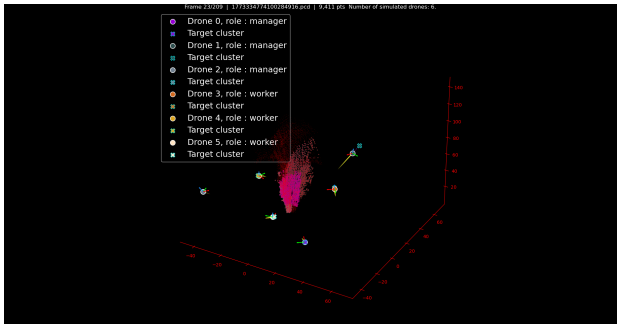
Consequently, the decision was made to use a simpler approach, albeit one that is less adaptable to real-world situations. Given that the plume of smoke is the only dynamic object in the scene, each point cloud can be compared to a reference map created beforehand using data of the surrounding environment that excludes the smoke.

The reference map is the sum of the first point clouds of a sequence, without the smoke, and then converted into an octree grid. Then, for each new point cloud of the sequence, we generate a new octree which is compared to the reference map. All the voxels of the octree that were not here in the reference map are considered to contain points of smoke. To avoid outliers, a statistical outlier removal filter is applied to the remaining point cloud. Figure 5 shows two octree grids, one with and the other without the smoke. The voxels corresponding to the smoke are inside the red shape in figure 5b. The extraction of smoke can be applied in real time during the experimentations or afterwards using the C++ PCL library [28].

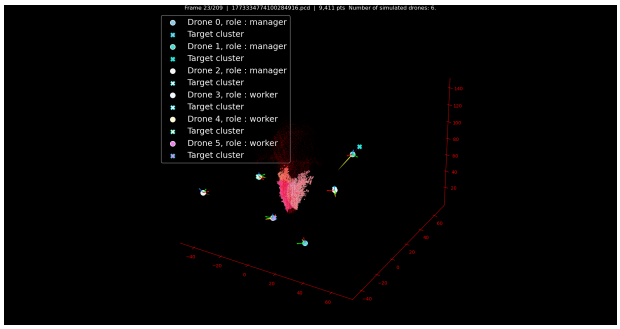
4 CASE STUDY : SWARM DRONE GUIDANCE

To demonstrate the usefulness of this data in the context of a drone swarm, we present an implementation of drone swarm guidance based on 3D point clouds recorded using the setup described in this paper. We aim to simulate a case of a drone swarm observing a smoke plume with a LiDAR sensor and reacting to its evolution.

http://www.imavs.org/



(a) Point clouds seen by each manager drone (each colour corresponds to one drone) and the original point cloud in transparent red.



(b) Point clouds seen by each worker drone (each colour corresponds to one drone) following a circle around the centroid point computed by managers and the original point cloud in transparent red

Figure 6: Manager and worker drones organised to observe the smoke plume. Each circle represents a drone.

Simulated LiDAR. In order to replicate a real-world scenario as closely as possible, in which each drone has its own incomplete perception of the smoke, a basic LiDAR simulation is integrated into each drone. To create this simulated LiDAR, the real specifications of the Livox MID-360 LiDAR are used to create a volume where the drone can perceive the point cloud. We arbitrarily decide to keep only the 30% closest points in order to not detect the points behind the surface, as it is the case with a real LiDAR. *Guidance*. Following the idea of [23], we organise the fleet with one manager group and one worker group, each one composed of 3 drones. The first group aims to give a global description of the smoke plume, computing its centroid and main directions of propagation based on their LiDAR observation. To do so, they are set to a static position facing towards the smoke (assuming that the approximate plume position is already known) and far from the plume (but still in the LiDAR range) to have a global view of the scene. Each drone computes the centroid position of their perceived point cloud with their simulated LiDAR, as described previously. Then, the mean of computed centroids is calculated and sent to the worker drones group. In figure 6a, the manager drones are displayed joining their positions to observe the smoke plume and compute the centroid. When receiving their target, the workers will join a circular formation and rotate around the smoke, facing towards the centroid at a closer distance than the managers. In figure 6, the worker drones are added; they circle around the centre of the smoke plume calculated by the manager drones. Each colour displayed on the smoke plume corresponds to a worker drone's perception.

5 CONCLUSION AND FUTURE WORKS

We presented in this paper an experimental setup to record synchronised and spatially calibrated smoke images and point clouds in a controlled environment, with an application to drone swarm control. We believe that this data can be useful for smoke recognition and drone control around smoke plumes, and a simple example of application is described enabling to better understand our view. The objective of this dataset is also to provide data that can be used for smoke modelling and behaviour learning.
Current works are looking to complete this dataset with a larger range of situations, from a controlled environment with adjusted lighting to an outside environment with natural conditions (wind, sunlight, dynamic background, ...), and developing maximum swarm drone coverage of a smoke plume using the presented data.

ACKNOWLEDGMENTS

The authors acknowledge the support of the French National Research Agency (Agence Nationale de la Recherche, ANR) under reference ANR-23-IAS2-0002-01 for the FireFlies TSIA ANR Project.

REFERENCES

[1] Brauer M. Johnston F. H. Jerrett M. Balmes J. R. Elliott Reid, C. E. Critical review of health impacts of wildfire smoke exposure. *Environmental health perspectives*, 2016.

[2] François Pimont Julien Ruffault Eric Rigolot, Jean-Luc Dupuy. Les incendies de forêt catastrophiques. *Annales des mines - Série Responsabilité et environnement*, 2020.

[3] Sharad Chitlangia Akhil Agnihotri Ding Zhao Hanjiang Hu, Zuxin Liu. Investigating the impact of multi-lidar placement on object detection for autonomous driving, 2022.

[4] Jianhao Jiao, Haoyang Ye, Yilong Zhu, and Ming Liu. Robust odometry and mapping for multi-lidar systems with online extrinsic calibration. *IEEE Transactions on Robotics*, 38(1):351–371, 2022.

[5] Kim Gon-Woo Sualeh Muhammad. Dynamic multi-lidar based multiple object detection and tracking. *Sensors*, 19(6), 2019.

http://www.imavs.org/

http://www.imavs.org/

[6] Murcia Moreno Harold. *Characterisation of Trees in Orchards using 3D LiDAR Data Collected by Mobile Robots*. Theses, Université de Toulouse, December 2025.

[7] Jimenez-Berni & al. High throughput determination of plant height, ground cover, and above-ground biomass in wheat with lidar. *Frontiers in Plant Science*, Volume 9 - 2018, 2018.

[8] Escolà Alexandre & al. Mobile terrestrial laser scanner applications in precision fruticulture/horticulture and tools to extract information from canopy point clouds. *Precision Agriculture*, 18(1):111–132, Feb 2017.

[9] Lin Cao Guibin Wang Fuliang Cao Kun Liu, Xin Shen. Estimating forest structural attributes using uav-lidar data in ginkgo plantations. *ISPRS Journal of Photogrammetry and Remote Sensing*, 146:465–482, 2018.

[10] Xiaoxia Zhao & al. Analysis of uav lidar information loss and its influence on the estimation accuracy of structural and functional traits in a meadow steppe. *Ecological Indicators*, 135:108515, 2022.

[11] Cheng Zhao Dexin Yao Guanyin Chen Li Sun Tomas Krajnik Zhi Yan Tao Yang, You Li. 3d tof lidar in mobile robotics: A review, 2022.

[12] Mcaree Peter Phillips Tyson, Guenther Nicky. When the dust settles: The four behaviors of lidar in the presence of fine airborne particulates. *Journal of Field Robotics*, 34, 02 2017.

[13] C Stiller R Urtasun A Geiger, P Lenz. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[14] I Grant. Particle image velocimetry: A review. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 211(1):55–76, 1997.

[15] Tim Hawkins, Per Einarsson, and Paul Debevec. Acquisition of time-varying participating media. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, page 812–815, New York, NY, USA, 2005. Association for Computing Machinery.

[16] Christian Fuchs, Tongbo Chen, Michael Goesele, Holger Theisel, and Hans-Peter Seidel. Volumetric Density Capture From a Single Image. In Raghu Machiraju and Torsten Moeller, editors, *Volume Graphics*. The Eurographics Association, 2006.

[17] Wieneke Bernhard Oudheusden Bas Elsinga Gerrit, Scarano Fulvio. Tomographic particle image velocimetry. *Experiments in Fluids*, 41:933–947, 12 2006.

[18] Wolfgang Heidrich Art Tevs Derek Bradley Marcus Magnor Hans-Peter Seidel Bradley Atcheson, Ivo Ihrke. Time-resolved 3d capture of non-stationary gas flows. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 27(5):132, 2008.

[19] James Gregson, Ivo Ihrke, Nils Thuerey, and Wolfgang Heidrich. From capture to simulation: connecting forward and inverse problems in fluids. *ACM Trans. Graph.*, 33(4), July 2014.

[20] Marie-Lena Eckert, Kiwon Um, and Nils Thuerey. Scalarflow: a large-scale volumetric data set of real-world scalar transport flows for computer animation and machine learning. *ACM Trans. Graph.*, 38(6), November 2019.

[21] Yue Gao, Hong-Xing Yu, Bo Zhu, and Jiajun Wu. Fluidnexus: 3d fluid reconstruction and prediction from a single video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2025.

[22] N. P. Lareau and C. B. Clements. Environmental controls on pyrocumulus and pyrocumulonimbus initiation and development. *Atmospheric Chemistry and Physics*, 16(6):4005–4022, 2016.

[23] Nikil Krishnakumar, Shashank Sharma, Srijan Kumar Pal, and Jiarong Hong. 3d characterization of smoke plume dispersion using multi-view drone swarm, 2025.

[24] Klapa Przemysław Mitka Bartosz. Procedure and accuracy assessment results of livox mid-360 scanners. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLVIII-1/W6-2025:147–154, 12 2025.

[25] Ieee standard for a precision clock synchronization protocol for networked measurement and control systems. *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pages 29–34, 2008.

[26] Klein Reinhard Schnabel Ruwen, Wahl Roland. Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, volume 26, pages 214–226. Wiley Online Library, 2007.

[27] Mersch Benedikt Wiesmann Louis Gall Jürgen Behley Jens Stachniss Cyrill Chen Xieyuanli, Li Shijie. Moving object segmentation in 3d lidar data: A learning-based approach exploiting sequential data. *IEEE Robotics and Automation Letters*, 6(4):6529–6536, 2021.

[28] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011. IEEE.

Vision-guided Velocity Control of Micro Air Vehicles using Rate-decoded Spiking Neural Network

Yufei Peng, Ni Li, and Hann Woei Ho*
School of Aeronautics, Northwestern Polytechnical University,
and National Key Laboratory of Aircraft Configuration Design, 710072 xi'an, China

http://www.imavs.org/

ABSTRACT

Micro air vehicles (MAVs) operate under strict size, weight, power, and computational constraints, which make heavy deep neural controllers challenging for onboard deployment. Spiking neural networks (SNNs) and event-based vision have recently emerged as promising approaches for energy-efficient aerial autonomy. Existing studies mainly focus on low-level control or direct event-driven sensing, while the generation of high-level velocity commands from visual observations remains insufficiently explored. This paper proposes a rate-decoded SNN imitation-learning controller for vision-guided MAV velocity control. The proposed controller maps compact visual-state observations, including target offsets, apparent scale, and motion cues, to continuous three-axis translational velocity commands using leaky integrate-and-fire neurons and spike-rate decoding. The controller is trained through imitation learning from a privileged PID expert that has access to vehicle position and velocity feedback, whereas the SNN receives only visual observations. The training objective combines behavioral cloning, command-smoothness regularization, and spike-rate regularization. Experimental results in a visual target-approaching task show that the learned SNN reproduces the dominant target-following behavior of the expert controller, continuously reduces tracking error, and maintains stable spiking activity during closed-loop execution. The results demonstrate the feasibility of compact rate-decoded SNNs for high-level vision-guided MAV control.

1 INTRODUCTION

Micro Air Vehicles (MAVs) are increasingly expected to perform autonomous navigation and decision-making tasks under strict constraints in size, weight, power, and onboard computation. This makes efficient perception and control a central challenge for sustained autonomous flight. Recent studies have demonstrated that autonomous MAVs can perform highly dynamic tasks, such as deep-learning-based MAV racing and real-time visual navigation [2]. However, deploying computation-intensive neural controllers on resource-constrained MAV platforms remains challenging in practice. Therefore, perception and control algorithms for MAVs must satisfy stringent requirements in terms of computational overhead, memory footprint, inference latency, and energy consumption [3,4]. These constraints make the design of lightweight and energy-efficient neural control architectures a central challenge for enabling practical autonomous MAV flight.

To address the computational and energy constraints of MAV autonomy, spiking neural networks (SNNs) have emerged as a promising alternative to conventional Artificial Neural Networks (ANNs). SNNs are commonly regarded as the third generation of neural network models, where information is represented by temporally sparse spike events rather than continuous-valued activations [5]. By maintaining internal neuronal states, such as membrane potentials and synaptic currents, SNNs perform event-driven temporal computation and transmit information only when firing thresholds are reached. This sparse communication mechanism makes SNNs naturally compatible with event-based sensors, temporal sensory streams, and neuromorphic processors. Moreover, surrogate-gradient learning enables SNNs to be optimized with gradient-based methods while preserving discrete spike-based forward dynamics [6, 7]. Because SNNs rely on sparse event-driven computation, they can reduce computational overhead compared with dense activation-based neural models, which make them suitable for resource-constrained onboard MAV systems with limited latency, memory, and energy budgets.

Recent studies have demonstrated the potential of SNNs for MAV control. Fully neuromorphic vision-to-control systems have shown that raw event-camera data can be processed by SNNs to generate low-level control commands for autonomous MAV flight [8]. Neuromorphic attitude estimation and control has further shown that modular SNNs can map raw inertial measurements to motor-related control commands on an MAV platform [9]. In addition, fully spiking actor–critic control trained with proximal policy optimization has been applied to high-speed continuous MAV control [10]. These studies demonstrate the feasibility of SNN-based MAV autonomy, but existing methods mainly focus on low-level attitude control, event-camera-based control, or reinforcement-

*Email: hannwoei@nwpu.edu.cn

http://www.imavs.org/

learning-based agile flight. For MAV visual servoing, it remains necessary to investigate whether a compact SNN policy can learn high-level velocity commands from visual observations while maintaining stable closed-loop tracking behavior.

In this work, we propose an SNN-based imitation learning framework for high-level MAV velocity control. The proposed method focuses on designing a compact spiking neural controller that maps low-dimensional visual observations to translational velocity commands. The controller is implemented using leaky integrate-and-fire neurons, which allow temporal information to be represented through internal neuronal states and sparse spike activity. Given the visual observation vector, the SNN generates bounded velocity outputs along the x, y, and z axes for closed-loop target approaching. The SNN controller is trained through imitation learning from a privileged PID expert. The expert has access to vehicle position and velocity feedback and provides reference translational velocity commands, whereas the SNN learns to approximate the expert policy using only compact visual observations. A rate-decoding mechanism is adopted to convert spike activity into continuous control commands. To improve control accuracy and temporal stability, the training objective combines behavioral cloning loss, command-jerk regularization, and spike-rate regularization. This design encourages the SNN to produce accurate, smooth, and biologically plausible spiking control outputs. Experimental results demonstrate that the trained SNN can reproduce the main target-approaching behavior and achieve stable closed-loop visual tracking.

The main contributions of this work are summarized as follows:

- A rate-decoded leaky integrate-and-fire SNN controller is proposed for high-level MAV velocity control, which enables continuous three-axis translational velocity generation from compact visual observations.
- An imitation-learning framework is introduced that combines expert-command supervision, temporal command-smoothness regularization, and spike-rate regularization to improve tracking accuracy, control smoothness, and spiking stability.
- Closed-loop simulation experiments demonstrate that the proposed controller reproduces the dominant target-approaching behavior of a physical-feedback expert while maintaining stable spiking dynamics during visual servoing.

2 METHODOLOGY

2.1 Pipeline Overview

Fig. 1 illustrates the overview of the proposed closed-loop SNN imitation-learning pipeline for vision-guided MAV velocity control. The pipeline consists of four main components: visual observation, expert demonstration and imitation training, the rate-decoded SNN controller, and the control output applied to the MAV. The simulation environment provides vehicle odometry and visual target observations for closed-loop interaction. During training, a conventional expert controller computes reference translational velocity commands from the physical tracking error, while the SNN learns to predict corresponding velocity commands from compact visual observations. In the current training configuration, the expert command is used to drive the vehicle, and the SNN is optimized in parallel through supervised imitation. During evaluation, the trained SNN policy is deployed as the controller and directly sends its predicted velocity commands to the vehicle velocity-control interface.

The task is formulated as vision-guided target approaching for an MAV. The vehicle is required to approach a visual reference target and maintain stable tracking behavior through translational velocity control. Instead of using high-dimensional image inputs, the controller receives a compact observation vector obtained from the visual target feature stream:

$$\mathbf{o}_t = [q_t, d_{x,t}, d_{y,t}, s_t, g^e_{x,t}, g^e_{y,t}]^\top. \tag{1}$$

Here, q_t denotes the marker-identifier channel provided by the visual feature message, $d_{x,t}$ and $d_{y,t}$ denote the normalized horizontal and vertical image-plane offsets of the visual target, and s_t represents the normalized apparent target scale. The terms $g^e_{x,t}$ and $g^e_{y,t}$ denote the horizontal and vertical components of the time-surface motion vector. The resulting six-dimensional feature vector is directly used as the SNN observation. In the implementation, the horizontal image coordinate is inverted to match the vehicle lateral-control direction. The image-plane offsets, apparent target scale, and motion-vector components are normalized and clipped to bounded ranges to improve numerical stability and reduce the influence of outlier visual measurements during training.

The SNN policy maps the observation vector $\mathbf{o}_t$ to a bounded translational velocity command:

$$\mathbf{v}^{\mathrm{cmd}}_t = [v^{\mathrm{cmd}}_{x,t}, v^{\mathrm{cmd}}_{y,t}, v^{\mathrm{cmd}}_{z,t}]^\top. \tag{2}$$

Throughout this paper, $\mathbf{v}^{\mathrm{cmd}}_t$ denotes the velocity command predicted by the SNN, and $\mathbf{v}^{\mathrm{E}}_t$ denotes the reference velocity command generated by the expert controller. The yaw-rate command is fixed to zero in the current setting; therefore, the learning problem focuses on three-axis translational velocity control. The SNN controller consists of a continuous feature encoder, a leaky integrate-and-fire spiking layer, and a linear readout layer. Spike activity is accumulated over multiple internal simulation steps and converted into continuous control outputs through rate decoding. The decoded command is further bounded by a hyperbolic tangent function and axis-wise velocity limits of $[0.35, 0.35, 0.25]^\top$ m/s for the x, y, and z axes, respectively, to match the expert command range.

The privileged PID expert generates reference velocity commands using vehicle position and velocity feedback,

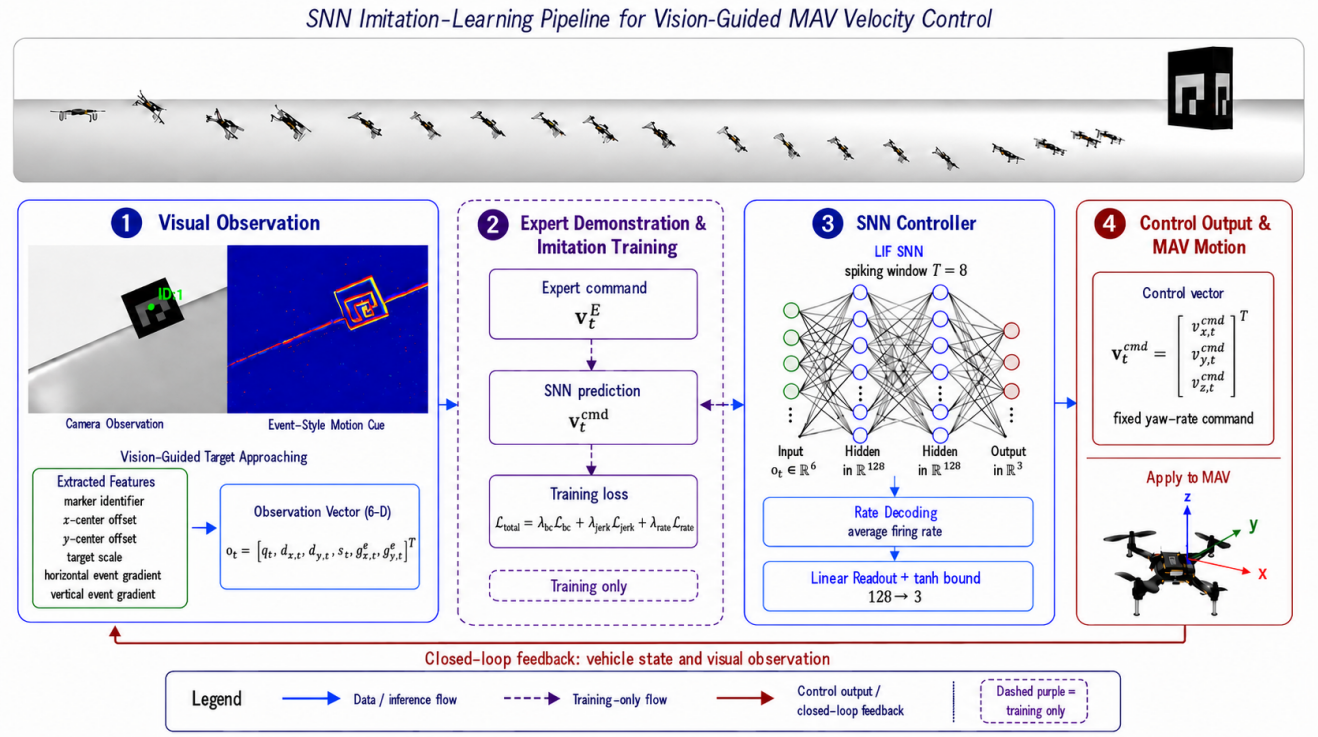


Figure 1: Overview of the proposed closed-loop imitation-learning pipeline. During training, the expert controller provides reference velocity commands for supervised imitation, while the rate-decoded SNN learns to map compact visual observations to three-axis velocity commands. During evaluation, the trained SNN is deployed as the closed-loop velocity controller.

which are not available to the SNN policy. These expert commands serve as supervised targets for the SNN. The training objective combines axis-weighted behavioral cloning loss, command-jerk regularization, and spike-rate range regularization. The behavioral cloning term encourages the SNN output to match the expert velocity command, the jerk term penalizes abrupt temporal variations in the predicted command sequence, and the spike-rate term constrains the firing activity within a desired range. This formulation enables the SNN controller to learn accurate, smooth, and stable spiking control outputs for closed-loop target approaching.

2.2 *Implementation of Spiking Neural Networks*

The proposed control policy is implemented as a rate-decoded spiking neural network (SNN) composed of leaky integrate-and-fire (LIF) neurons. Unlike conventional artificial neural networks that transmit continuous activations at each layer, the LIF-based SNN maintains an internal temporal state through membrane-potential dynamics and communicates information using binary spike events. Such temporal dynamics allow the controller to encode short-term variations in visual observations while preserving the event-driven computational characteristics of spiking neural networks.

The policy maps a compact visual observation vector $\mathbf{o}_t$ to a continuous translational velocity command. The observation vector contains the normalized visual and motion-related features extracted at control step t. The overall network consists of an input projection module, an LIF spiking computation layer, a rate-decoding module, and a continuous velocity readout. First, the observation is embedded into a hidden feature representation through a multilayer projection function:

$$\mathbf{h}_t = f_{\mathrm{proj}}(\mathbf{o}_t), \tag{3}$$

where $f_{\mathrm{proj}}(\cdot)$ is composed of fully connected layers with normalization and nonlinear activation. This projection transforms the low-dimensional observation into a latent representation suitable for subsequent spiking computation.

The projected feature $\mathbf{h}_t$ is then injected into the LIF spiking layer over a short internal time window of length T. In the reported experiments, the internal spiking window is set to $T = 8$. According to the implementation, the LIF layer contains 128 neurons, with $\tau_{\mathrm{mem}}^{-1} = 10.0$, $\tau_{\mathrm{syn}}^{-1} = 5.0$, and a firing threshold of $\vartheta_{\mathrm{th}} = 0.30$. At each internal spiking step k, the LIF neurons update their membrane potential according to the incoming current and their previous hidden state. When the membrane potential exceeds the firing threshold, a binary spike is generated and the neuron state is reset. The spike-generation process can be written as

$$\boldsymbol{\sigma}_{t,k}, \boldsymbol{\xi}_{t,k} = f_{\mathrm{LIF}}\left(\mathbf{h}_t, \boldsymbol{\xi}_{t,k-1}\right), \quad k = 1, \ldots, T, \tag{4}$$

where $\boldsymbol{\sigma}_{t,k}$ denotes the binary spike vector at the k-th internal

http://www.imavs.org/

step, and $\boldsymbol{\xi}_{t,k}$ represents the corresponding LIF neuron state, including the membrane potential and synaptic current. The generated spike sequence within one control step is denoted as

$$\mathcal{B}_t = \{\boldsymbol{\sigma}_{t,1}, \boldsymbol{\sigma}_{t,2}, \ldots, \boldsymbol{\sigma}_{t,T}\}. \tag{5}$$

This internal temporal unfolding enables the policy to convert a compact visual feature into a spike-based temporal representation before producing the final velocity command.

Since the quadrotor requires continuous velocity commands, a rate-decoding strategy is adopted to bridge the discrete spike representation and the continuous control space. Specifically, the spike-rate vector is computed by averaging the binary spike sequence over the internal time window:

$$\bar{\boldsymbol{\sigma}}_t = \frac{1}{T}\sum_{k=1}^{T} \boldsymbol{\sigma}_{t,k}. \tag{6}$$

The decoded spike-rate vector $\bar{\boldsymbol{\sigma}}_t$ provides a continuous representation of the spiking activity and is used as the input to the final velocity readout layer. The translational velocity command is obtained by

$$\mathbf{v}_t^{\mathrm{cmd}} = \mathbf{v}_{\max} \odot \tanh\left(\mathbf{W}_o \bar{\boldsymbol{\sigma}}_t + \mathbf{b}_o\right), \tag{7}$$

where $\mathbf{W}_o$ and $\mathbf{b}_o$ are the learnable parameters of the linear readout layer, $\tanh(\cdot)$ bounds the output range, $\mathbf{v}_{\max}$ denotes the maximum allowable velocity for each axis, and $\odot$ represents element-wise multiplication.

The final output of the policy is defined as

$$\mathbf{v}_t^{\mathrm{cmd}} = \left[v_{x,t}^{\mathrm{cmd}}, v_{y,t}^{\mathrm{cmd}}, v_{z,t}^{\mathrm{cmd}}\right]^{\top}, \tag{8}$$

which corresponds to the translational velocity commands along the x, y, and z axes. In the current setting, the yaw-rate command is not learned by the SNN and is fixed to zero during policy execution. Different from approaches that rely on an additional output low-pass filter, the proposed policy directly uses the rate-decoded SNN output as the control command. Therefore, the temporal smoothness of the generated velocity command is mainly determined by the intrinsic LIF dynamics, the spike-rate decoding mechanism, and the smoothness-aware training objective.

2.3 Expert-Guided Imitation Learning

The proposed SNN controller is trained using expert-guided imitation learning. During training, an expert policy is used to provide reference translational velocity commands for the visual servoing task. The expert policy is implemented as a privileged PID-based outer-loop velocity controller, which computes stable velocity demonstrations from vehicle position and velocity feedback according to the tracking error between the vehicle and the target. These expert commands serve as supervised labels for training the SNN policy.

At each control step, the perception module extracts the visual observation $\mathbf{o}_t$, and the SNN predicts a velocity command $\mathbf{v}_t^{\mathrm{cmd}}$ from this observation. In parallel, the expert policy generates the reference command $\mathbf{v}_t^{\mathrm{E}}$. The SNN is optimized to imitate the expert command while preserving smooth and functional spiking behavior. During the early training stage, the simulator is driven by the expert command rather than the SNN output. This prevents unstable commands from an untrained SNN from destabilizing the vehicle and allows the model to learn from stable closed-loop trajectories.

Algorithm 1 Closed-loop rate-decoded SNN imitation learning

```
Initialize environment, expert, and SNN policy
for episode = 1 to 1000 do
    Randomize the initial MAV pose
    Reset expert integrator and SNN hidden state
    Set rollout length by curriculum schedule
    for step = 1 to T_ep do
        Extract and normalize visual observation o_t
        Compute expert command v_t^E
        Predict SNN command v_t^cmd by rate decoding
        Accumulate losses after warm-up steps
        Apply the expert command to the simulator
        if step is divisible by 20 then
            Detach SNN hidden state from the graph
        end if
    end for
    Backpropagate the accumulated loss and update SNN parameters
end for
```

The training process follows a closed-loop rollout procedure. For each episode, the vehicle is initialized from a randomized pose, the expert and SNN hidden states are reset, and the system is rolled out for a fixed number of control steps. After an initial warm-up period, the SNN prediction and expert command are collected to compute the imitation loss. The SNN hidden state is periodically detached from the computational graph to avoid excessively long backpropagation histories while preserving its temporal state during execution. After each episode, the accumulated loss is backpropagated through the SNN using surrogate gradients, and the network parameters are updated with gradient-based optimization.

The rollout length follows a curriculum schedule from 100 to 600 control steps. For an episode of length T_{ep}, the warm-up duration is set to $N_{\mathrm{w}} = \min(50, \lfloor 0.3T_{\mathrm{ep}} \rfloor)$ control steps. These initial steps are excluded from loss accumulation to reduce the influence of reset transients in the LIF state and visual-motion features. Adam optimization is used with a learning rate of 10^{-4} and weight decay of 10^{-6}.

2.4 Loss Function

The SNN policy is trained by minimizing a composite loss that combines expert imitation, command smoothness,

http://www.imavs.org/

and spike-activity regularization. The total loss is defined as

$$\mathcal{L}_{\text{total}} = \lambda_{\text{bc}}\mathcal{L}_{\text{bc}} + \lambda_{\text{jerk}}\mathcal{L}_{\text{jerk}} + \lambda_{\text{rate}}\mathcal{L}_{\text{rate}}. \tag{9}$$

In the reported experiments, the loss weights are set to $\lambda_{\text{bc}} = 1.0$, $\lambda_{\text{jerk}} = 0.02$, and $\lambda_{\text{rate}} = 0.01$. An additional task-direction metric is computed during training for monitoring purposes, but its weight is set to zero and it is not used for parameter optimization.

The primary learning objective is axis-weighted behavioral cloning, which penalizes the discrepancy between the SNN command $\mathbf{v}_t^{\text{cmd}}$ and the expert command $\mathbf{v}_t^{\text{E}}$:

$$\mathcal{L}_{\text{bc}} = \frac{1}{3|\mathcal{T}|}\sum_{t\in\mathcal{T}}\sum_{j\in\{x,y,z\}} w_j \left(v_{j,t}^{\text{cmd}} - v_{j,t}^{\text{E}}\right)^2, \tag{10}$$

where $\mathcal{T}$ denotes the set of training steps after the warm-up period, and the axis-weight vector is set to $\mathbf{w} = [1.0, 6.0, 3.0]^\top$. Larger weights are assigned to the lateral and vertical axes because these channels are more sensitive to visual-servoing errors and were observed to be more difficult to fit during training.

To suppress abrupt command variations, a command-jerk regularization term is introduced. It penalizes the second-order temporal difference of the predicted velocity commands over a rollout:

$$\mathcal{L}_{\text{jerk}} = \frac{1}{T_{\text{ep}} - 2}\sum_{t=3}^{T_{\text{ep}}} \left\|\mathbf{v}_t^{\text{cmd}} - 2\mathbf{v}_{t-1}^{\text{cmd}} + \mathbf{v}_{t-2}^{\text{cmd}}\right\|_2^2. \tag{11}$$

This term encourages temporally smooth velocity outputs without applying an explicit low-pass filter to the final command.

The spike-rate regularization term is designed to maintain functional spiking activity in the hidden LIF layer. Instead of enforcing a fixed target firing rate, a range-based penalty is used:

$$\mathcal{L}_{\text{rate}} = \left[\max\left(0, r_{\text{low}} - \bar{r}\right)\right]^2 + \left[\max\left(0, \bar{r} - r_{\text{high}}\right)\right]^2, \tag{12}$$

where $\bar{r}$ denotes the mean spike rate of the hidden LIF layer. The lower and upper bounds are set to $r_{\text{low}} = 0.05$ and $r_{\text{high}} = 0.18$, respectively. This formulation prevents silent neurons and saturated firing activity while allowing the network to learn a flexible range of effective spike rates.

3 EXPERIMENTAL SETUP

The experiments are conducted in a ROS2-Gazebo simulation environment based on the Crazyflie nano-quadrotor platform. Following the software-in-the-loop simulation idea of CrazySim [12], the environment is implemented using ROS2, Gazebo, and the open-source Crazyflie ROS2-Gazebo package[1]. The package provides a Gazebo-based Crazyflie model, ROS2 communication interfaces, and velocity-command interaction, enabling repeatable closed-loop evaluation of vision-guided MAV control.

The perception and control pipeline is implemented as ROS2 nodes. The onboard camera stream is processed using OpenCV, and an ArUco marker is used as the visual reference. Compact visual features, including marker center, apparent scale, and short-term image-space motion cues, are extracted and used as the input to the SNN controller. The SNN is implemented with PyTorch and Norse [11], where leaky integrate-and-fire neurons and surrogate-gradient-compatible spiking computation are used. Velocity commands are published to the Crazyflie simulation interface, while odometry feedback is obtained through the ROS2-Gazebo bridge.

The task is defined as vision-guided target approaching in a three-dimensional workspace. A floating visual reference marker is placed at $[2.2, 0, 0.8]^\top$ m, while the desired target position is set to $\mathbf{p}^* = [2.0, 0, 0.8]^\top$ m. This offset reflects a practical visual-servoing setting, where the marker provides perception guidance and the UAV is expected to stabilize at a desired relative position rather than directly collide with the marker. At the beginning of each rollout, the UAV is initialized from a randomized position sampled from $x_0 \in [-2.0, 0.0]$ m, $y_0 \in [-1.0, 1.0]$ m, and $z_0 \in [0.75, 0.95]$ m. The range is chosen to ensure that the marker is visible in most initial states while still providing sufficient variation in distance and lateral offset.

For evaluation, the trained SNN controller and the expert reference controller are tested from the same initial poses over a fixed rollout length of 600 control steps. The comparison considers 3D trajectory, tracking error, per-axis velocity command, command jerk, and spike activity. Command jerk is computed from the discrete second-order difference of the velocity command, while spike activity is measured by the mean hidden-layer spike rate. These metrics jointly evaluate target-reaching performance, control smoothness, and the functional spiking behavior of the proposed controller.

4 EXPERIMENTAL RESULTS

This section evaluates the learned SNN controller in the vision-guided target-approaching task. The expert controller uses physical position and velocity feedback, whereas the SNN controller generates three-axis velocity commands from compact visual observations. Therefore, the evaluation focuses on whether the SNN can reproduce the dominant target-approaching behavior rather than exactly match the expert trajectory at every time step.

Fig. 2 presents a representative individual trajectory produced by the rate-decoded SNN controller. The stroboscopic sequence in panel (a) visualizes the complete target-approaching motion. Panels (b) and (c) show the trajectory projected onto the x–z and x–y planes, respectively, with color encoding the elapsed time. Starting from the light-colored square, the MAV progresses toward the target while

[1] https://github.com/bitcraze/crazyflie-simulation

http://www.imavs.org/

http://www.imavs.org/

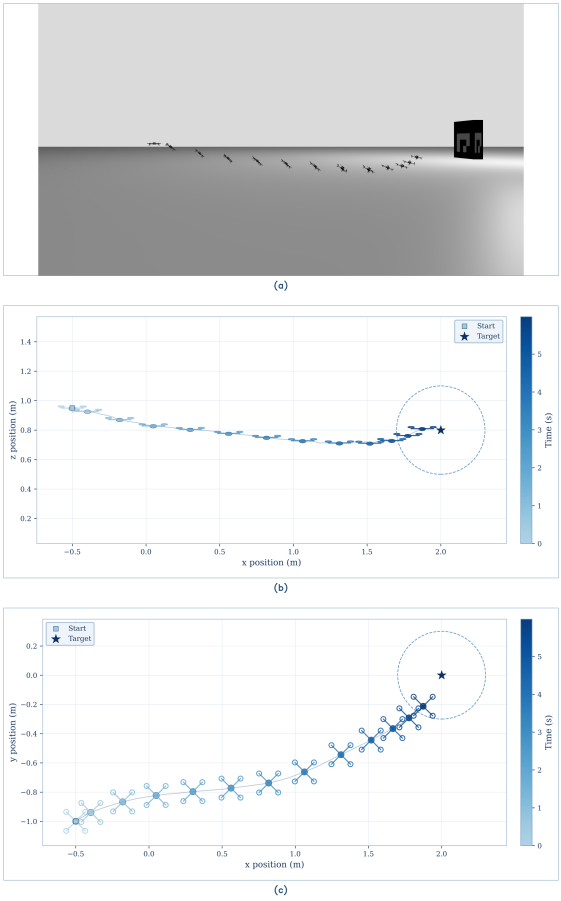


Figure 2: Visualization of the MAV target-approaching trajectory and convergence behavior.

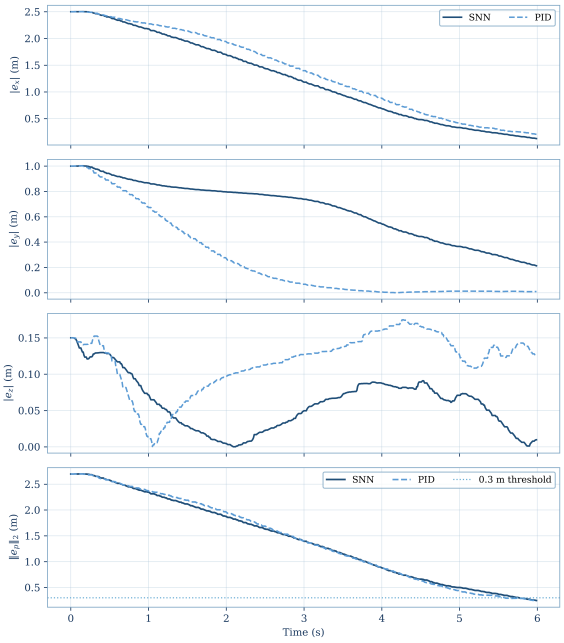


Figure 3: Position tracking error comparison over the evaluation rollout.

correcting its lateral position and altitude. The trajectory converges toward the dashed target region, demonstrating stable target-approaching behavior during closed-loop visual servoing.

The tracking-error curves in Fig. 3 show that both controllers continuously reduce the initial position error during the 600-step rollout. The SNN follows the same decreasing trend as the expert, indicating that the learned controller can generate effective closed-loop velocity commands from visual observations.

Fig. 4 reports the position responses along the x, y, and z axes. The x-axis response shows that both controllers move consistently toward the target, indicating that the SNN learns the dominant forward target-approaching behavior. The y-axis response indicates that the SNN can reduce the lateral deviation, although its convergence profile differs from that of the expert controller. The SNN z-axis response remains bounded around the target altitude but shows a larger residual offset than the expert response. The vertical dashed line indicates the time step at which the camera temporarily loses the visual target. This loss of visual reference may reduce the reliability of the compact visual observation and contribute to the subsequent lateral and vertical deviations. These results suggest that the SNN learns a usable visual servoing policy, while axis-level precision is still affected by visual-target visibility and closed-loop dynamics.

The velocity-command comparison in Fig. 5 further explains the observed motion. Along the x axis, the SNN produces a forward velocity close to the dominant expert command. Along the y and z axes, the SNN commands are less adaptive and show local fluctuations. Nevertheless, the generated commands remain sufficient to guide the MAV toward the target region.

Command smoothness is evaluated using discrete command jerk, as shown in Fig. 6. The SNN produces a short initial jerk peak after rollout begins, but the jerk remains low for most of the trajectory. Since no output low-pass filter is applied, this result suggests that the rate-decoding mechanism and smoothness-aware training provide usable command continuity.

Fig. 7 shows the hidden-layer spike activity during SNN evaluation. The spike rate increases at the beginning and then remains stable, indicating that the LIF layer stays active during closed-loop control and that the velocity command is produced through functional spike-rate representations.

To evaluate robustness under randomized initial conditions, the SNN controller was tested from different randomly sampled initial positions, as shown in Fig. 8. The resulting trajectories show that the SNN consistently drives the MAV toward the target region from different initial states. The

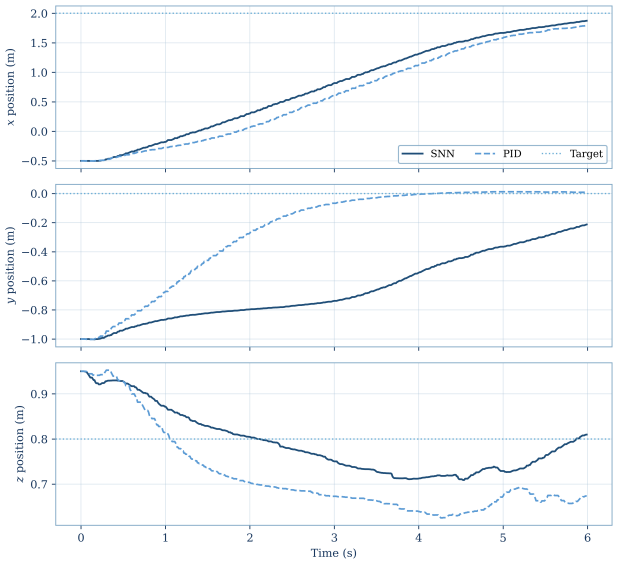


Figure 4: Position-axis comparison for the expert controller and the SNN controller.

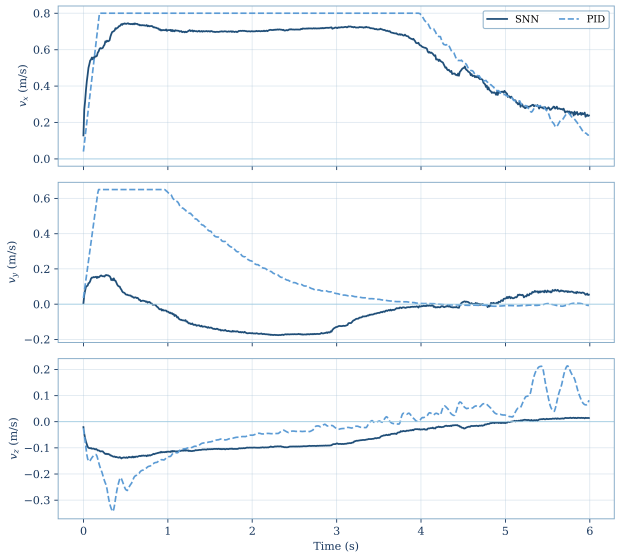


Figure 5: Velocity command comparison along the x, y, and z axes.

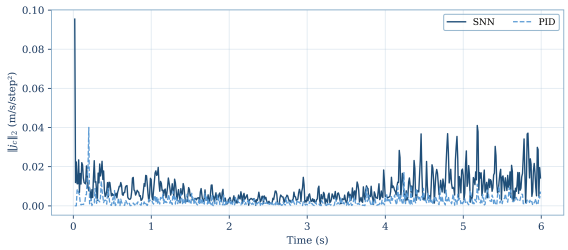


Figure 6: Command-jerk comparison between the expert controller and the SNN controller.

trajectories exhibit clear forward progress along the x direction, lateral correction from both sides of the workspace, and bounded vertical motion around the target altitude with small residual offsets. These results demonstrate that the learned controller preserves a consistent target-approaching tendency under randomized initial conditions.

Overall, the experimental results show that the rate-decoded SNN controller can reproduce the main visual target-approaching behavior of the expert controller. The learned policy reduces tracking error, generates bounded velocity commands, and maintains stable spike activity during closed-loop execution. Remaining lateral and vertical deviations indicate that axis-level precision can be further improved through better visual-state representation, smoother readout dynamics, and closed-loop fine-tuning.

5 Conclusion

This paper presented a rate-decoded SNN controller for vision-guided velocity control of an MAV. The proposed method uses compact visual-state observations as input and learns continuous three-axis translational velocity commands by imitating a privileged PID expert with access to vehicle position and velocity feedback. The SNN policy is built with leaky integrate-and-fire spiking dynamics, where hidden spike activity is converted into continuous control outputs through spike-rate decoding. To improve learning stability and control quality, the training objective combines axis-weighted behavioral cloning, command-jerk regularization, and spike-rate regularization. Experimental results demonstrate that the learned SNN controller can reproduce the dominant target-approaching behavior of the expert controller. The vehicle is driven toward the target region, the tracking error is gradually reduced, and the overall convergence trend is comparable to the expert reference. The spike-activity analysis further validates that the hidden spiking layer remains active during closed-loop execution, which indicates that the controller preserves functional spiking dynamics while producing continuous velocity commands. Overall, the results show that rate-decoded SNN imitation learning provides an effective and compact control framework for vision-guided MAV target approaching. Future work will extend the framework toward attitude-aware control and richer motion-cue representations to better handle coupled translational and rotational visual dynamics, together with onboard deployment on resource-constrained hardware.

References

[1] O. K. Pal, "In-depth review of AI-enabled unmanned aerial vehicles," *Innovative Infrastructure Solutions*, vol. 9, Art. no. 415, 2024.

[2] L. Lamberti, E. Cereda, G. Abbate, L. Bellone, V. J. K. Morinigo, M. Barcis, A. Barcis, A. Giusti, F.

http://www.imavs.org/

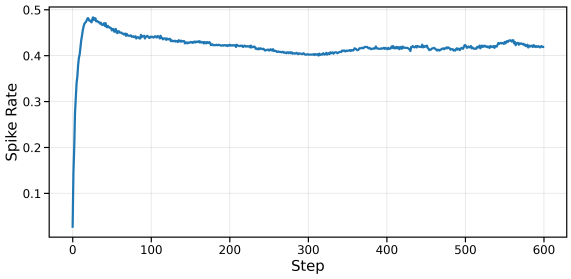


Figure 7: Average hidden-layer spike activity during SNN evaluation.

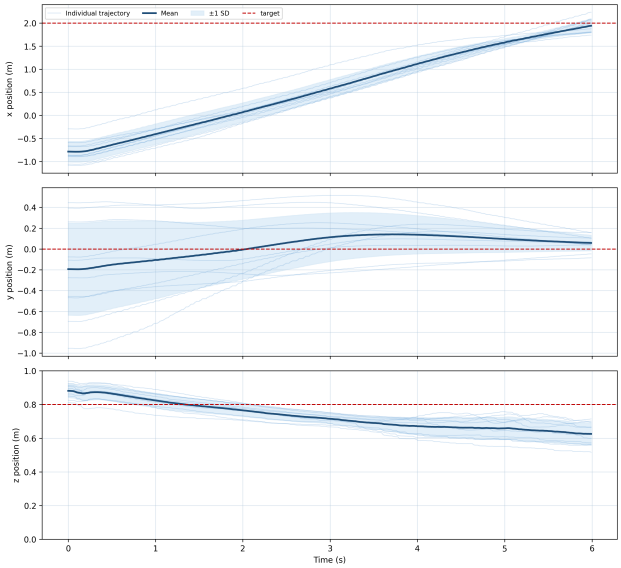


Figure 8: SNN trajectories under randomized initial conditions.

http://www.imavs.org/

Conti, and D. Palossi, "A sim-to-real deep learning-based framework for autonomous nano-drone racing," *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 1899–1906, 2024.

[3] C. Vincent, L. Hirsch, N. Ul Hasan, C. C. E. Fernandes, A. Crainic, J. Zasada-James, and J. Baldwin, "Real-time edge intelligence in UAV for search-and-rescue: Onboard energy-efficient video summarisation with reduced data transmission," in *Proceedings of the 15th International Conference on the Internet of Things*, ACM, pp. 52–58, 2025.

[4] A. Abdalla, M. M. A. Mohammed, O. Adedeji, P. Dotray, and W. Guo, "Toward resource-efficient UAV systems: Deep learning model compression for onboard-ready weed detection in UAV imagery," *Smart Agricultural Technology*, vol. 12, Art. no. 101086, 2025.

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

[6] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019, doi: 10.1109/MSP.2019.2931595.

[7] J. K. Eshraghian, M. Ward, E. O. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong, and W. D. Lu, "Training spiking neural networks using lessons from deep learning," *Proceedings of the IEEE*, vol. 111, no. 9, pp. 1016–1054, 2023, doi: 10.1109/JPROC.2023.3308088.

[8] F. Paredes-Valles, J. J. Hagenaars, J. Dupeyroux, S. Stroobants, Y. Xu, and G. C. H. E. de Croon, "Fully neuromorphic vision and control for autonomous drone flight," *Science Robotics*, vol. 9, no. 90, Art. no. eadi0591, 2024, doi: 10.1126/scirobotics.adi0591.

[9] S. Stroobants, C. De Wagter, and G. C. H. E. de Croon, "Neuromorphic attitude estimation and control," *IEEE Robotics and Automation Letters*, vol. 10, no. 5, pp. 4858–4865, 2025, doi: 10.1109/LRA.2025.3553418.

[10] M. B. van Breukelen Castillo, C. De Wagter, R. Ferede, R. Vos, and G. C. H. E. de Croon, "Spiking neural networks for high-speed continuous quadcopter control using proximal policy optimization," in *Proceedings of the 16th International Micro Air Vehicle Conference and Competition*, San Andres Cholula, Mexico, 2025, pp. 133–141.

[11] C. Pehle and J. E. Pedersen, "Norse – A deep learning library for spiking neural networks," Zenodo, version 0.0.7, 2021, doi: 10.5281/zenodo.4422025.

[12] C. Llanes, Z. Kakish, K. Williams, and S. Coogan, "CrazySim: A software-in-the-loop simulator for the Crazyflie nano quadrotor," in *Proceedings of the 2024 IEEE International Conference on Robotics and Automation*, Yokohama, Japan, 2024, pp. 12248–12254, doi: 10.1109/ICRA57147.2024.10610906.

Decentralized PSO-Based Guidance for Multi-UAV Wildfire Monitoring and Frontier Tracking

H. Chakraa, M. Feurgard, and M. Bronz
École Nationale de l'Aviation Civile (ENAC), Université de Toulouse, 31400 Toulouse, France.

http://www.imavs.org/

Abstract

This paper proposes a guidance framework for adaptive navigation and coverage control in autonomous Unmanned Aerial Vehicles (UAVs) swarms operating in rapidly evolving wildfire environments. In the navigation phase, each UAV computes the bearing to its nearest discovered frontier cell and advances along that heading, with an iterative separation to prevent inter-drone collisions. In the coverage phase, a decentralized Particle Swarm Optimization (PSO) based controller searches the velocity space independently for each drone. Its fitness simultaneously maximizes front-line coverage, rewards exploration of the pre-frontier region just ahead of the fire edge, and attracts each drone toward its assigned angular sector of the perimeter. A shared belief fire map, updated incrementally through exploration, drives both frontier extraction and the PSO fitness. We evaluate the framework with frontier and fire-body Intersection-over-Union (IoU), sensor overlap rate, and per-UAV decision latency. Finally, the proposed framework was experimentally validated through real-world flight tests using a fleet of DJI Mini 3 quadrotors.

1 Introduction

Wildfires rank among the most destructive natural disasters, causing irreversible ecological damage, economic losses, and severe threats to human lives and infrastructures [1]. The increasing frequency and intensity of wildfires highlight the urgent need for efficient monitoring technologies. Unmanned Aerial Vehicles (UAVs) have emerged as a promising solution, offering higher spatial and temporal resolution and enabling access to inaccessible terrain [2].

The central challenge is guidance in a dynamic and partially observed environment. A wildfire front evolves continuously under wind, terrain, and fuel conditions [3]. Effective fleet coordination involves two complementary subproblems: (i) *navigation*, guiding drones from initial positions to the active fire, and (ii) *coverage and tracking*, distributing drones along the frontier boundary to maximize observability, following the propagation (see Figure 1).

Emails: {hamza.chakraa, mael.feurgard, murat.bronz}@enac.fr

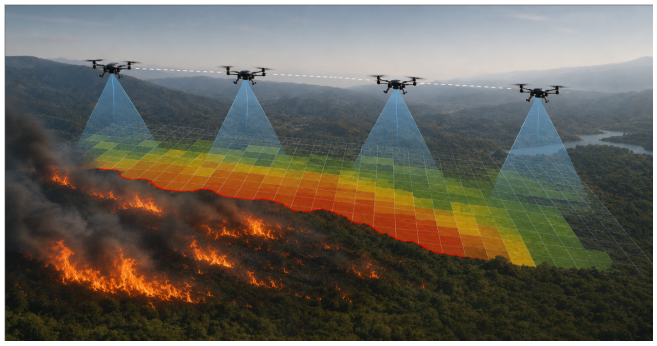


Figure 1: Illustration of a wildfire monitoring scenario. UAVs cooperatively track the evolving wildfire frontier. Each drone observes a local region within its onboard sensor range, and then information is constructed.

This paper addresses this problem through a unified guidance framework. The proposed approach operates in two sequential phases. In the navigation phase, each UAV is driven toward the evolving wildfire frontier using a heading-based controller designed for partial observability, where no prior map of the fire is available. The swarm incrementally builds the situation through onboard sensing, progressively revealing the fire structure as the mission progresses. Once sufficient proximity to the wildfire is achieved, the system transitions to the coverage phase, where the objective shifts from reaching the fire to spatially distributing UAVs along its boundary. This phase is formulated as a decentralized Particle Swarm Optimization (PSO) problem [4], enabling each UAV to independently optimize its velocity while remaining coordinated at the swarm level. PSO has been widely adopted in UAV deployment and coverage tasks due to its balance between computational efficiency and global search capability [5]. A key component of this work is a frontier extraction module based on OpenCV image-processing techniques, which continuously reconstructs the wildfire boundary from the shared belief fire map.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 presents the system overview and models. Section 4 details the multi-UAV guidance strategy. Section 5 reports simulation results and field demonstrations with DJI Mini 3 drones, and finally, Section 6 concludes.

http://www.imavs.org/

2 RELATED WORK

Satellite imagery and ground-based sensor networks provide large-scale coverage, but their low temporal resolution limits real-time response [5]. UAV platforms bridge this gap with on-demand deployment at higher spatial and temporal resolution, including inaccessible terrains to ground personnel [6, 7]. Bailon-Ruiz *et al.* [6] introduced an early multi-UAV replanning system for wildfire monitoring, later extended to continuous real-time perimeter tracking [8]. Heracleous *et al.* [7] demonstrated accurate frontier reconstruction from aerial imagery, and Seraj *et al.* [9] formalized cooperative coverage with quality of service and safety guarantees. Despite this progress, guidance under partial observability (where the fire boundary must be discovered incrementally) remains a central open challenge.

Classical Coverage Path Planning (CPP) problems assume a known, static target region and produce offline tours that guarantee complete coverage with bounded redundancy. The wildfire setting fundamentally breaks this assumption: the region of interest is the active fire perimeter, which is unknown *a priori*, partially observable, and continuously deforming under wind and fuel gradients [8]. Coverage must therefore be treated as an online dynamic problem in which the target itself moves and grows, sensor information arrives incrementally, and replanning occurs at every control step. This shift, from offline CPP over a fixed map to adaptive tracking of an evolving frontier, motivates dedicated dynamic formulations and distinguishes wildfire guidance from conventional multi-UAV area coverage.

A complementary thread frames wildfire monitoring as an active perception problem, where agents maximize information gain about the fire state rather than cover fixed geometric regions. Julian and Kochenderfer [10] showed that Deep Reinforcement Learning (DRL) with information-driven objectives can outperform geometric coverage heuristics in decentralized surveillance. Wang and Bronz [11] proposed a Monte Carlo Tree Search (MCTS) belief-state planner for uncertainty estimation, fire tracking, and fuel-aware exploration. These methods show strong estimation performance but require either large training datasets or expensive online tree search. Evolutionary strategies such as Genetic Algorithms [12] and PSO offer a training-free alternative that remains competitive for deployment. PSO in particular performs strongly in decentralized multi-agent settings: its population-based search and support for multi-objective fitness make it well-suited for online re-optimization.

To the best of our knowledge, this is the first framework to apply per-drone decentralized PSO as a reactive velocity controller for wildfire frontier coverage. Unlike prior approaches that rely on discrete action sets [6, 8], offline distillation [11], or trained networks [13, 14], our framework is training-free, interpretable, and runs on a single CPU without a GPU or pre-collected data.

3 SYSTEM OVERVIEW

3.1 Wildfire propagation model

The wildfire environment is represented by a discrete two-dimensional grid $\mathcal{G} = \{1, \dots, N\}^2$ where each cell corresponds to a spatial region of the terrain. The simulation evolves in discrete time steps. At each time step, The fire state is $\mathbf{X}_t \in [0,1]^{N\times N}$ evolving with a fuel map $\mathbf{F}_t \in \mathbb{R}_{\geq 0}^{N\times N}$. Wildfire propagation follows a Cellular Automata (CA) formulation on the Moore neighborhood. Full model details are in [15]. Fire is propagated every $N_f = 10$ drone control steps, yielding an effective ground-truth spread rate of approximately 0.5–$1\,\mathrm{m/s}$, consistent with measured rates for low-to-moderate intensity fires [8].

3.2 UAVs and observation model

We consider a fleet of M homogeneous UAVs operating over the fire grid. Each UAV i has a planar pose $\mathbf{p}_t^{(i)} = (x_t^{(i)}, y_t^{(i)}) \in \mathbb{R}^2$ and a heading $\theta_t^{(i)} \in (-\pi, \pi]$. The sensor footprint $\mathcal{D}_t^{(i)}$ of drone i at time t is a square field of view of side s cells centered at the grid projection of $\mathbf{p}_t^{(i)}$:

$$\mathcal{D}_t^{(i)} = \left\{ \tilde{\mathbf{p}} \in \mathcal{G} \;\middle|\; \left\| \left[\tilde{\mathbf{p}} - \mathrm{m2cell}(\mathbf{p}_t^{(i)})\right] \right\|_\infty \leq \rho \right\} \quad (1)$$

where $\rho = \lfloor s/2 \rfloor$ is the FoV half-side in cells and $\mathrm{m2cell}(\cdot)$ denotes the coordinate transformation. Within $D_t^{(i)}$, drone i observes the true fire state $\mathbf{X}_t(\tilde{\mathbf{p}})$.

The team maintains a single shared data structure: the belief fire map $\hat{\mathbf{X}}_t \in [0,1]^{N\times N}$, initialized to zero. The belief map is updated incrementally as the fleet explores: after each control step, every cell inside the union of all drone FoV footprints is overwritten with its ground-truth fire value, while unobserved cells retain their previous belief:

$$\hat{\mathbf{X}}_{t+1}(\tilde{\mathbf{p}}) = \begin{cases} \mathbf{X}_t(\tilde{\mathbf{p}}) & \text{if } \tilde{\mathbf{p}} \in \bigcup_{i=1}^{M} \mathcal{D}_t^{(i)}, \\ \hat{\mathbf{X}}_t(\tilde{\mathbf{p}}) & \text{otherwise.} \end{cases} \quad (2)$$

This update rule models the physical sensing process: cells that no drone has yet overflown remain at their initialized value while observed cells hold the most recent ground-truth measurement. The belief frontier $\hat{\mathbf{R}}_t$ is extracted from $\hat{\mathbf{X}}_t$ after every observation update via the OpenCV pipeline described in Section 3.3.

3.3 Frontier extraction

The active fire frontier is extracted using a multi-step OpenCV pipeline designed to construct the estimated fire frontier, $\hat{\mathbf{R}}_t$, from the shared belief fire map $\hat{\mathbf{X}}_t$. The process begins by smoothing isolated burning pixels and suppressing artifacts that typically appear near the boundary. After this initial denoising stage, fragmented fire regions created by the blurring operation are reconnected using binary thresholding at $\tau = 60$ and morphological refinement. Residual artifact

http://www.imavs.org/

pixels are then removed using a median filter, while an accumulation pass fills small holes that would otherwise generate false interior contours.

Once a coherent fire mask has been obtained, the boundary is further regularized using Gaussian blur to reduce high-frequency irregularities before edge detection. Finally, the active fire frontier is extracted as a thin single-pixel contour using the Canny edge detector [16].

The resulting binary edge image $\mathcal{E}_t \in \{0,1\}^{N\times N}$ encodes detected frontier pixels, where $\mathcal{E}_t(\tilde{\mathbf{p}}) = 1$ indicates an edge and $\mathcal{E}_t(\tilde{\mathbf{p}}) = 0$ indicates background. The estimated frontier is then:

$$\hat{\mathbf{R}}_t = \{\tilde{\mathbf{p}} \in \mathcal{G} \mid \mathcal{E}_t(\tilde{\mathbf{p}}) = 1\} \tag{3}$$

The same pipeline is applied to the ground-truth fire map to obtain $\mathbf{R}_t$, used *exclusively* for metrics calculation (Section 5).

4 MULTI-UAV GUIDANCE STRATEGY

Before coverage can begin, each drone must reach the active fire frontier. The system operates under complete initial ignorance: $\hat{\mathbf{X}}_0 = \mathbf{0}_{N\times N}$ and $\hat{\mathbf{R}}_0 = \emptyset$, so no frontier is known at $t = 0$ and drones have no prior information about the fire location. Drones are initially dispatched to the four quadrant centroids of the operational area. As soon as any drone's FoV intersects a burning region and $\hat{\mathbf{R}}_t \neq \emptyset$, the transition to coverage occurs.

Once all UAVs have reached the frontier and the system switches to the coverage phase, each one runs an independent PSO instance at every control step to determine its velocity command. The only information shared between drones is their current positions $\{\mathbf{p}_t^{(j)}\}_{j\neq i}$ and the shared belief map $\hat{\mathbf{X}}_t$, both available to drone i. Before running the PSO, two preliminary operations are performed: sector assignment, which defines the portion of the frontier that drone i is responsible for, and the boundary check, which determines whether the PSO should run at all or whether the drone should instead return toward its assigned region. These are described in the following.

4.1 Sector assignment

To prevent fleet clustering, the belief frontier $\hat{\mathbf{R}}_t$ is partitioned into M equal angular sectors around its centroid $\bar{\mathbf{f}}_t$. Each frontier pixel is assigned to the sector whose angular slice contains its bearing from $\bar{\mathbf{f}}_t$, dividing the perimeter into M arcs of equal angular width $2\pi/M$. The sector centroid $\mathbf{s}^{(i)}$ for drone i is then the mean position of all frontier pixels in its assigned arc:

$$\mathbf{s}^{(i)} = \frac{1}{|\mathcal{S}^{(i)}|} \sum_{\mathbf{q}\in\mathcal{S}^{(i)}} \mathbf{q} \quad \in \mathbb{R}^2 \tag{4}$$

where $\mathcal{S}^{(i)}$ is the set of positions of frontier pixels belonging to drone i's sector, falling back to $\bar{\mathbf{f}}_t$ if the sector is empty. Sector assignment is recomputed at every control step as the belief frontier evolves. If drone i drifts farther than the effective leash radius ℓ_{eff} from $\mathbf{s}^{(i)}$, the PSO is bypassed entirely and the drone returns directly toward its sector centroid. The leash radius adapts to the current frontier size so that drones are allowed to move farther as the fire grows:

$$\ell_{\text{eff}} = \max(\ell_0,\ 1.4 \cdot \bar{r}_{\mathcal{F}}) \tag{5}$$

where ℓ_0 is the base leash and $\bar{r}_{\mathcal{F}}$ is the mean distance from belief frontier pixels to $\bar{\mathbf{f}}_t$ in metres, serving as a proxy for the current frontier radius. A fixed leash would cause boundary-pinning as the frontier expands: drones would be repeatedly recalled before they can cover the outer portions of a large perimeter.

4.2 PSO update

A population of P particles evolves over K iterations, each particle k encoding a candidate drone velocity $\boldsymbol{\xi}_k = (v_x^k, v_y^k)^\top$ constrained to the speed disc $\mathcal{B}(V) = \{\mathbf{v} \in \mathbb{R}^2 \mid \|\mathbf{v}\|_2 \leq V\}$, which enforces a physical speed limit. The search first updates the particles internal search velocity $\boldsymbol{\nu}_k \in \mathbb{R}^2$:

$$\begin{aligned}\boldsymbol{\nu}_{k+1} = w\,\boldsymbol{\nu}_k \\ + c_1\mathbf{r}_1 \odot (\boldsymbol{\xi}_k^{\text{pb}} - \boldsymbol{\xi}_k) \\ + c_2\mathbf{r}_2 \odot (\boldsymbol{\xi}^{\text{gb}} - \boldsymbol{\xi}_k)\end{aligned} \tag{6}$$

where $\boldsymbol{\xi}_k^{\text{pb}}$ is the personal best (the highest-fitness candidate this particle has visited across all previous iterations) and $\boldsymbol{\xi}^{\text{gb}}$ is the global best (the highest-fitness candidate found by any particle in the population). $\mathbf{r}_1$ and $\mathbf{r}_2$ are independent random vectors with each component drawn uniformly from $[0,1]$ at every iteration, and $\odot$ denotes element-wise multiplication. The second equation updates the candidate velocity itself:

$$\boldsymbol{\xi}_{k+1} = \text{proj}_{\mathcal{B}(V)}(\boldsymbol{\xi}_k + \boldsymbol{\nu}_{k+1}) \tag{7}$$

For temporal continuity, the population is warm-started at every control step: 80% of particles are initialized near the previous step's global best $\boldsymbol{\xi}^{\text{gb}}$ with additive Gaussian noise and the remaining 20% are sampled uniformly on $\mathcal{B}(V)$. This preserves knowledge of the previous solution while maintaining sufficient exploration to escape local optima.

4.3 Fitness function

The fitness Φ_i is evaluated independently for each of the P particles at every PSO iteration. For a particle k with candidate velocity $\boldsymbol{\xi}_k = (v_x^k, v_y^k)^\top$, the candidate position is:

$$\mathbf{p}'_k = \mathbf{p}_t^{(i)} + \boldsymbol{\xi}_k\,\mathrm{d}t \in \mathbb{R}^2 \tag{8}$$

and its grid projection is $\tilde{\mathbf{p}}'_k$. The fitness combines terms in two coordinate spaces: the coverage terms G_i and E_i use the grid projection $\tilde{\mathbf{p}}'_k$ to index into the belief map and count

http://www.imavs.org/

frontier pixels inside the FoV window, while the attraction A_i, separation C_i, and boundary B terms use the world position $\mathbf{p}'_k$ directly, since they involve physical distances for which the grid representation is inappropriate. The fitness score of a particle k with candidate velocity $\boldsymbol{\xi}_k$ is:

$$\begin{aligned}\Phi_i(\boldsymbol{\xi}_k) = &\underbrace{\lambda_G\, G_i(\tilde{\mathbf{p}}'_k)}_{\text{exclusive gain}} + \underbrace{\lambda_E\, E_i(\tilde{\mathbf{p}}'_k)}_{\text{exploratory gain}} \\ &+ \underbrace{\lambda_A\, A_i(\mathbf{p}'_k)}_{\text{sector attraction}} - \underbrace{\lambda_C\, C_i(\mathbf{p}'_k)}_{\text{collisions}} - \underbrace{\lambda_B\, B(\mathbf{p}'_k)}_{\text{boundary}}\end{aligned} \tag{9}$$

Exclusive coverage gain G_i: This term rewards the candidate position for covering frontier pixels that no teammate currently observes. The gain counts frontier pixels visible from $\mathbf{p}'_k$ that no teammate sees:

$$G_i(\tilde{\mathbf{p}}'_k) = \frac{1}{s^2}\left|\mathcal{D}(\tilde{\mathbf{p}}'_k) \cap \hat{\mathbf{R}}_t \cap \left(\mathcal{G} \setminus \bigcup_{j\neq i} \mathcal{D}_t^{(j)}\right)\right| \tag{10}$$

where $|\cdot|$ denotes set cardinality and $\mathcal{D}(\tilde{\mathbf{p}}'_k)$ denote the cell FoV window centered at the candidate grid position $\tilde{\mathbf{p}}'_k$. This single term is self-driving: a stationary drone exhausts its exclusive cells and is pushed outward; entering the fire body yields zero gain; only tangential motion along the ring continuously reveals new exclusive cells.

Exploratory gain E_i: When the frontier is fully covered, the exploratory gain extends each drone's attention beyond the current fire edge and into a pre-frontier zone $\mathcal{P}_t$ (the unburned cells that the fire has not yet reached but will reach next). It counts the cells in $\mathcal{P}_t$ visible from $\tilde{\mathbf{p}}'_k$ that no other UAV currently covers:

$$E_i(\tilde{\mathbf{p}}'_k) = \frac{1}{s^2}\left|\mathcal{D}(\tilde{\mathbf{p}}'_k) \cap \mathcal{P}_t \cap \left(\mathcal{G} \setminus \bigcup_{j\neq i} \mathcal{D}_t^{(j)}\right)\right| \tag{11}$$

and the pre-frontier zone is obtained by morphological expansion of the belief frontier followed by exclusion of already-burning cells:

$$\begin{aligned}\mathcal{P}_t = &\left\{\tilde{\mathbf{p}} \in \mathcal{G} \;\middle|\; \min_{\tilde{\mathbf{p}}' \in \hat{\mathbf{R}}_t} \|\tilde{\mathbf{p}} - \tilde{\mathbf{p}}'\|_\infty \leq n_s\right\} \\ &\setminus \left(\hat{\mathbf{R}}_t \cup \left\{\tilde{\mathbf{p}} \in \mathcal{G} \mid \hat{\mathbf{X}}_t(\tilde{\mathbf{p}}) > 0.5\right\}\right)\end{aligned} \tag{12}$$

The expansion radius n_s cells corresponds to next fire propagation steps giving drones enough anticipatory range to observe cells before they ignite.

Sector attraction A_i: When both G_i and E_i are zero (which occurs when the frontier is fully covered and the pre-frontier buffer is also saturated), the fitness Φ_i receives no signal from the coverage terms. In this condition, the PSO has no basis for preferring one direction over another, and drones may drift arbitrarily or cluster together. The sector attraction term prevents this by providing a continuous, direction-dependent signal that pulls each drone toward its assigned sector centroid $\mathbf{s}^{(i)}$:

$$A_i(\mathbf{p}'_k) = \frac{1}{1 + \dfrac{\|\mathbf{p}'_k - \mathbf{s}^{(i)}\|_2}{D}} \tag{13}$$

where D is the diagonal of the operational area in meters, used as a normalization factor.

Collision penalty C_i: This term penalizes candidate positions that would bring UAV i too close to any other one, discouraging the PSO from selecting velocities that create near-collisions. For each drone $j \neq i$, the penalty is zero when the candidate distance $\|\mathbf{p}'_k - \mathbf{p}_t^{(j)}\|_2$ exceeds the minimum separation d_{sep}:

$$C_i(\mathbf{p}'_k) = \sum_{j\neq i}\left[\max\Big(0,\; d_{\text{sep}} - \|\mathbf{p}'_k - \mathbf{p}_t^{(j)}\|_2\Big)\right]^2 \tag{14}$$

This means the penalty increases rapidly as the violation grows. The sum over all other agents means the penalty also scales with the number of simultaneous violations, strongly discouraging positions where drone i would be close to multiple drones at once.

Boundary repulsion B: This term penalizes candidate positions that approach the edges of the operational area. If drone i flies within one FoV half-width of any grid edge, part of its sensor footprint extends outside the modeled area, wasting sensing capacity. The penalty activates linearly as soon as the candidate enters this margin along any edge and increases toward the boundary.

Table 1: PSO fitness terms and weights

Term	Symbol	Weight
Individual gain	G_i	$\lambda_G = 1.00$
Exploratory gain	E_i	$\lambda_E = 0.35$
Sector attraction	A_i	$\lambda_A = 0.40$
Collision penalty	C_i	$\lambda_C = 0.25$
Boundary repulsion	B	$\lambda_B = 0.04$

Fitness weights: The five weights are summarized in Table 1. The coverage terms G_i and E_i carry the highest weights as

http://www.imavs.org/

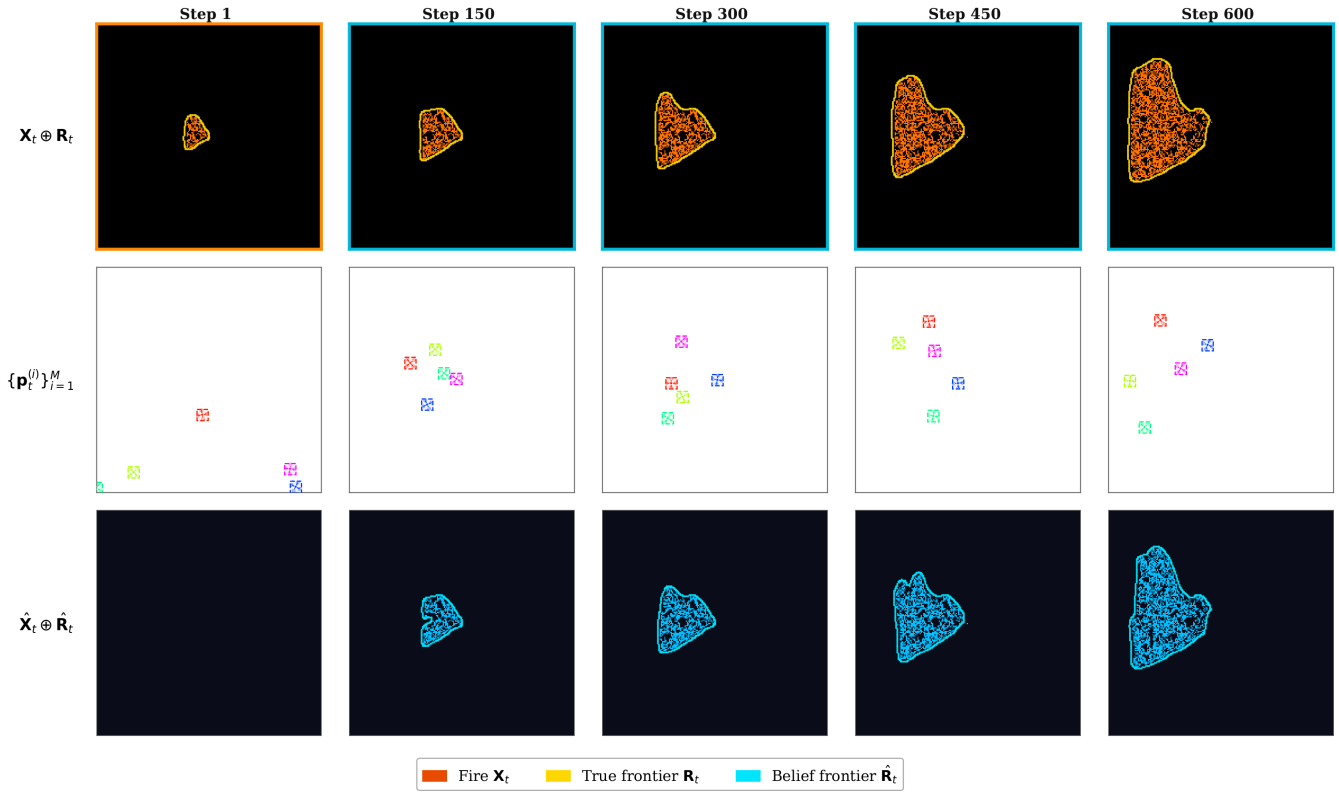


Figure 2: Simulation snapshot shown at five selected timesteps. *Row 1*: true fire state $\mathbf{X}_t$ overlaid with the ground-truth frontier $\mathbf{R}_t$ (yellow). *Row 2*: UAV positions $\{\mathbf{p}_t^{(i)}\}_{i=1}^{M}$ with their square FoV footprints. *Row 3*: shared belief map $\hat{\mathbf{X}}_t$ overlaid with the belief frontier $\hat{\mathbf{R}}_t$ (cyan), built incrementally from FoV observations. At $t = 0$, the belief map is entirely dark, reflecting complete initial ignorance $\hat{\mathbf{X}}_0 = \mathbf{0}$. As drones overfly burning regions, observed cells are overwritten with ground-truth values and the belief frontier grows, progressively narrowing the gap with the true frontier. The progression illustrates how the fleet collectively reconstructs the fire perimeter through exploration alone, with no prior map of the environment.

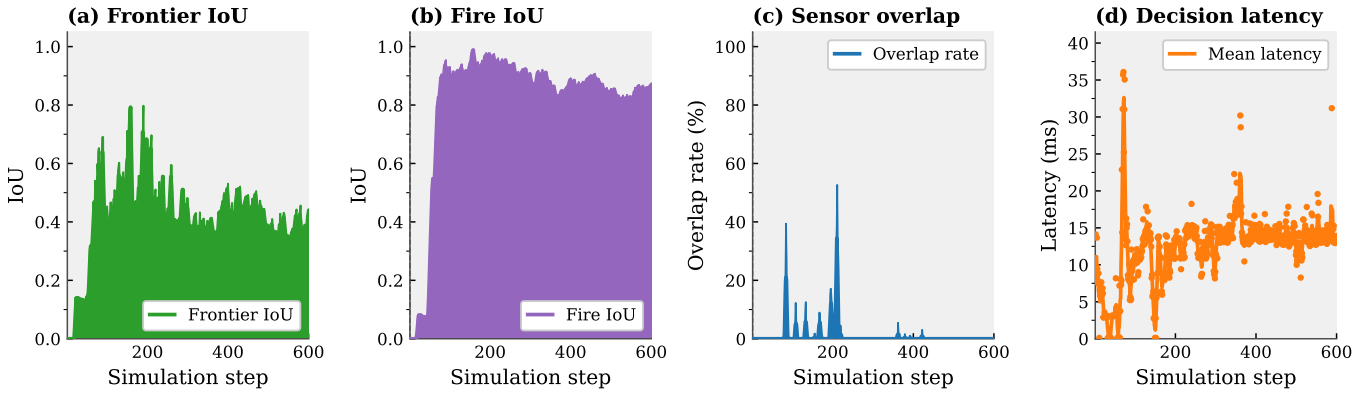


Figure 3: Per-step metrics over 600 simulation steps (200×200 grid). *(a)* Frontier IoU over the full run. *(b)* Fire IoU, reflecting coverage of the burning body. *(c)* Sensor overlap rate: low sustained value confirms efficient fleet distribution. *(d)* Per-drone PSO decision latency, well within the 1 s budget.

they are the primary drivers of cooperative behavior. The sector attraction A_i is weighted at $\lambda_A = 0.40$ to maintain distribution when coverage saturates, but also weak enough to be overridden by large G_i or E_i.

5 RESULTS AND FIELD DEPLOYMENT

We evaluate the framework in two scenarios: a large-scale simulation on a 200×200 grid, with 5 UAVs and real flights deployment on the ENAC football field with DJI Mini 3 quadrotors on a 100×100 grid with 3 UAVs.

5.1 Simulation

Scenario: We consider a 200×200 grid (cell size $\Delta = 5\,\mathrm{m}$, total area $1\,\mathrm{km}^2$) with $M = 5$ UAVs operating at $V = 5\,\mathrm{m/s}$ for simulation. Fire is ignited at the grid center and propagated every $N_f = 10$ steps, yielding an effective spread rate of approximately 0.5–1 $\mathrm{m/s}$, consistent with low to moderate intensity fires [8]. Full parameters are listed in Table 2.

Table 2: Simulation parameters

Type	Parameter	Value
Environment	Grid size $N \times N$	200×200
	Cell resolution Δ	$5\,\mathrm{m}$
	Fire update period N_f	10 steps
UAV	Cruise speed V	$5\,\mathrm{m/s}$
	Field of view s	10 cells
	Fleet size M	5
PSO guidance	Population size P	40
	Iterations K	30
	Inertia weight w	0.60
	Acceleration $c_1 = c_2$	1.50
	Separation distance d_{sep}	$40\,\mathrm{m}$

Results: Figure 2 shows the evolution of the simulation across selected timesteps. *Row 1* shows the true fire state $\mathbf{X}_t$ overlaid with the ground-truth frontier. *Row 2* shows the UAV positions $\{\mathbf{p}_t^{(i)}\}$ with FoV footprints. *Row 3* shows the shared belief map $\hat{\mathbf{X}}_t$ with the belief frontier in cyan, built incrementally from FoV observations. At $t = 0$ the belief map is entirely dark, reflecting complete initial ignorance $\hat{\mathbf{X}}_0 = \mathbf{0}$. As drones fly over the fire, observed cells are updated and the belief frontier narrows the gap with the ground-truth frontier.

Figure 3 reports four per-step metrics over the simulation steps. We evaluate the framework along four complementary dimensions. **Frontier Intersection over Union (IoU)** measures pixel-level agreement between the belief frontier $\hat{\mathbf{R}}_t$ and the ground-truth fire edge $\mathbf{R}_t$; it directly quantifies how accurately the fleet tracks the active fire boundary. **Fire IoU** measures the overlap between the belief fire body $\hat{\mathbf{X}}_t$ and the true burning area $\mathbf{X}_t$. It reflects how well the interior of the fire has been observed, complementing the boundary-focused frontier IoU. Further, **Sensor overlap rate** is the fraction of currently covered frontier pixels observed by two or more drones simultaneously, a low value indicates that the fleet is spatially distributed with minimal redundant sensing, which is the goal of the sector assignment mechanism. Finally, **Decision latency** is the wall-clock time spent by each drone computing its PSO velocity command, it is a direct indicator of real-time feasibility on a ground station without GPU acceleration.

5.2 Field deployment

Scenario: The framework was demonstrated on a fleet of $M = 3$ DJI Mini 3 quadrotors flying on the ENAC football field ($\approx 116\,\mathrm{m} \times 65\,\mathrm{m}$). The fire environment was simulated on a ground station using a 100×100 grid ($\Delta = 0.55\,\mathrm{m}$, with a total area of $55\,\mathrm{m} \times 55\,\mathrm{m}$), positioned as a square sub-region centred within the rectangular field boundary, as illustrated in Figure 4(c–d). Fire is ignited at the grid center and propagated every $N_f = 10$ steps. Drone sensing was simulated by the square FoV footprint $\mathcal{D}_t^{(i)}$ centred on each drone's GPS-reported position, so the shared belief map was updated from real flight telemetry. Velocity commands were forwarded to the flying drones via the WildBridge interface [17], which exposes the DJI Mobile SDK via HTTP and RTSP, following the architecture validated in prior work [18]. Guidance parameters for this scenario are $V = 2.5\,\mathrm{m/s}$, $s = 12$ cells ($6.6\,\mathrm{m}$), and $d_{\mathrm{sep}} = 15\,\mathrm{m}$.

Results: This setup validates the complete guidance pipeline under real flight conditions. On the field, the navigation phase was extremely short, drones encountered the simulated fire frontier within the first few seconds of flight and PSO coverage activated almost immediately. Figure 4 shows the drones flying on the ENAC football field during the demonstration, the DroneViz visualization (developed in ENAC) displays real-time drone positions, alongside a snapshot of the simulated fire state and the first 80 steps of GPS trajectories. Figure 5 shows the corresponding metrics.

The small operational area plays a decisive role in the observed performance. The FoV footprints collectively cover a substantial fraction of the grid at every step, so the belief map saturates rapidly and frontier IoU approaches unity within a few steps. Similarly, fire IoU reaches ≈ 1.0 almost immediately since the entire burning body falls within the observable area. These near-perfect scores therefore reflect the scale of the deployment rather than a fundamental improvement over the large-scale simulation.

6 CONCLUSION

In this work, we presented a multi-UAV guidance framework for autonomous wildfire frontier tracking. The proposed approach combines a navigation strategy with a decentralized velocity-space PSO controller designed for dynamic

http://www.imavs.org/

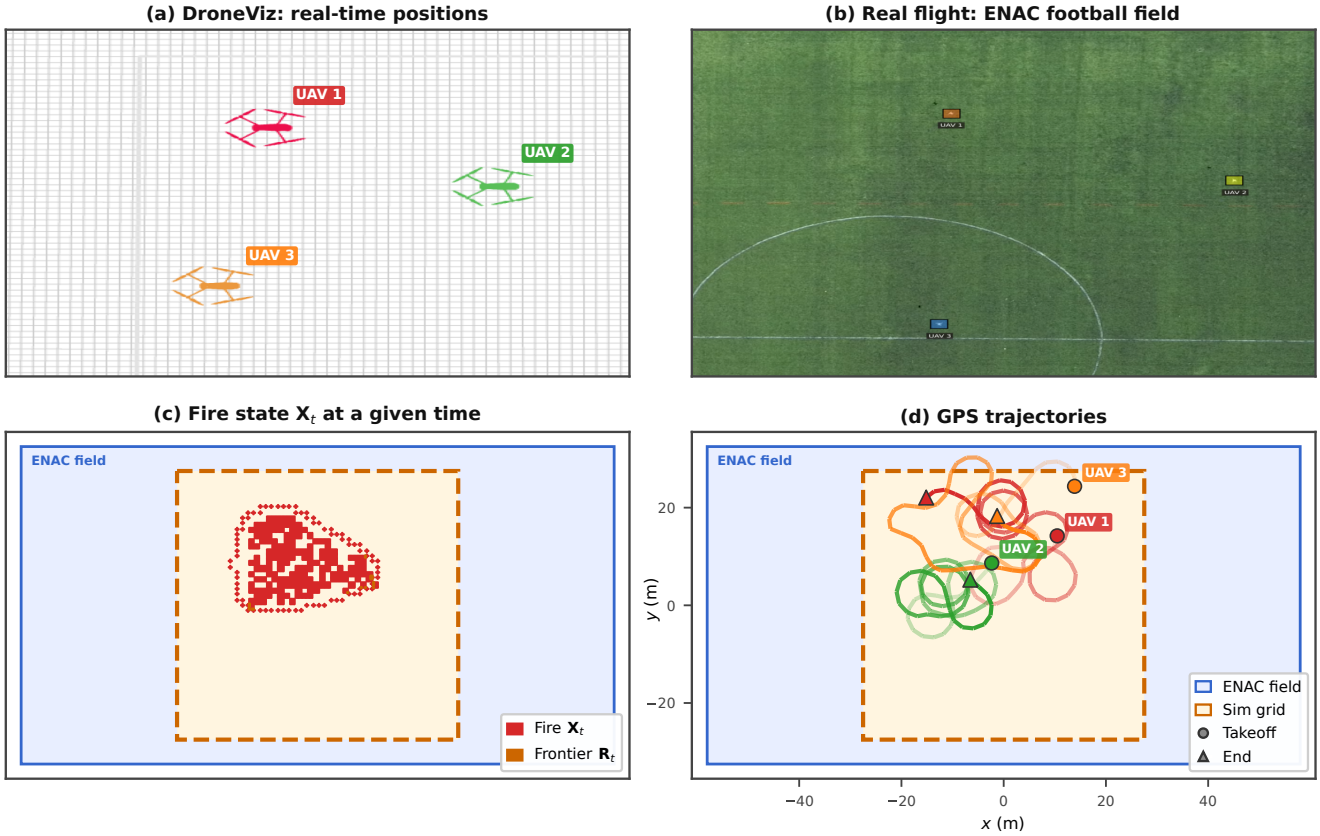


Figure 4: Field demonstration with $M = 3$ DJI Mini-3 quadrotors on the ENAC football field. Velocity commands from the guidance algorithm were forwarded via WildBridge framework. The fire environment and drone sensing were simulated on the ground station using a 100×100 grid ($\Delta = 0.55\,\mathrm{m}$).

coverage under partial observability. The fitness function integrates exclusive frontier gain, exploratory sensing, sector attraction, collision avoidance, and boundary-awareness terms to promote efficient and coordinated swarm behavior. In addition, the framework's interfaces enable straightforward deployment on real UAV systems without additional command translation layers. Future work will include extensive comparisons with state-of-the-art planning and learning-based approaches, robustness analyses under communication and sensing uncertainties. Finally, an extension to support heterogeneous teams combining fixed-wing and quadrotors will be considered.

Code and Data Availability

Code is available at: `https://github.com/drones-fireflies/pso-guidance`

Acknowledgments

The authors acknowledge the support of the French National Research Agency (Agence Nationale de la Recherche, ANR) under reference ANR-23-IAS2-0002-01 for the FireFlies TSIA ANR Project.

References

[1] Chi Yuan, Youmin Zhang, and Zhixiang Liu. A survey on technologies for automatic forest fire monitoring, detection, and fighting using unmanned aerial vehicles and remote sensing techniques. *Canadian Journal of Forest Research*, 45(7):783–792, 2015.

[2] Eladio Altamira-Colado, Daniel Cuevas-González, Marco A. Reyna, Juan Pablo García-Vázquez, Roberto L. Avitia, and Alvaro R. Osornio-Vargas. Drone-Assisted Particulate Matter Measurement in Air Monitoring: A Patent Review. *Atmosphere*, 15(5):515, 2024.

[3] Yuqin Wang, Fengsen Gao, and Minghui Li. Probabilistic Path Planning for UAVs in Forest Fire Monitoring: Enhancing Patrol Efficiency through Risk Assessment. *Fire*, 7(7):254, 2024.

[4] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN95 - International Conference on Neural Networks*, pages 1942–1948, 1995.

http://www.imavs.org/

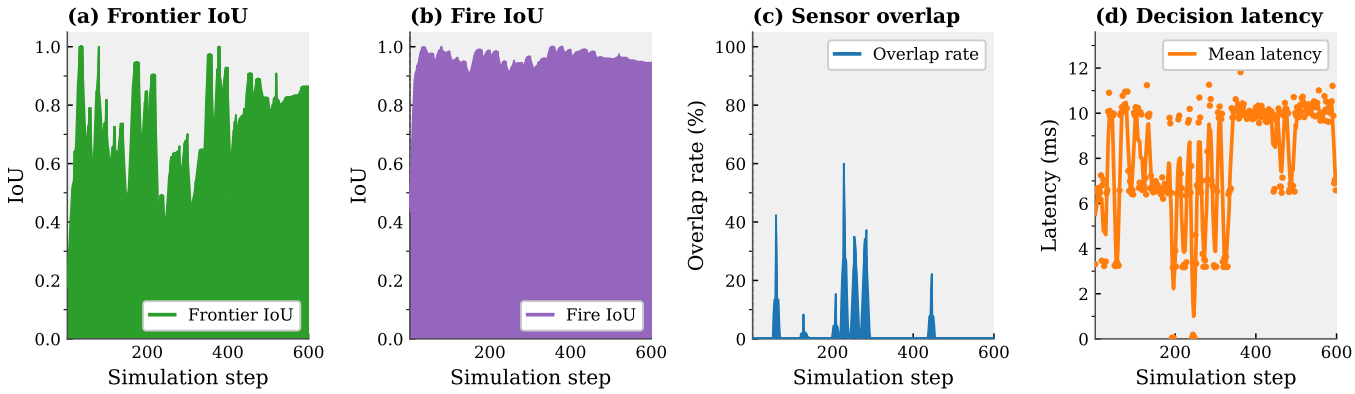


Figure 5: Per-step metrics in the field scenario with DJI Mini 3 quadrotors. *(a)* Frontier IoU over the full run. *(b)* Fire IoU, reflecting coverage of the burning body. *(c)* Sensor overlap rate. *(d)* Per-drone PSO decision latency

[5] Alessandro Giuseppi, Roberto Germanà, Federico Fiorini, Francesco Delli Priscoli, and Antonio Pietrabissa. UAV Patrolling for Wildfire Monitoring by a Dynamic Voronoi Tessellation on Satellite Data. *Drones*, 5(4):130, 2021.

[6] Rafael Bailon-Ruiz, Simon Lacroix, and Arthur Bit-Monnot. Planning to Monitor Wildfires with a Fleet of UAVs. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4729–4734, 2018.

[7] Constantinos Heracleous, Panayiotis Kolios, and Christos G. Panayiotou. UAV-based system for real-time wildfire perimeter propagation tracking. In *2023 31st Mediterranean Conference on Control and Automation (MED)*, pages 13–18, 2023.

[8] Rafael Bailon-Ruiz, Arthur Bit-Monnot, and Simon Lacroix. Real-time wildfire monitoring with a fleet of UAVs. *Robotics and Autonomous Systems*, 152:104071, 2022.

[9] Esmaeil Seraj, Andrew Silva, and Matthew Gombolay. Multi-UAV planning for cooperative wildfire coverage and tracking with quality-of-service guarantees. *Autonomous Agents and Multi-Agent Systems*, 36(2):39, 2022.

[10] Kyle D. Julian and Mykel J. Kochenderfer. Distributed Wildfire Surveillance with Autonomous Aircraft Using Deep Reinforcement Learning. *Journal of Guidance, Control, and Dynamics*, 42(8):1768–1778, 2019.

[11] Liyang Wang and Murat Bronz. Efficient decentralized multi-uav wildfire monitoring via mcts-distilled diffusion policies. In *14th International Conference on Unmanned Aircraft Systems (ICUAS)*, Corfu, Greece, 2026.

[12] Josy John and Suresh Sundaram. Genetic Algorithm-based Routing and Scheduling for Wildfire Suppression using a Team of UAVs. In *2024 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2024.

[13] Sahand Khoshdel, Qi Luo, and Fatemeh Afghah. PyroTrack: Belief-Based Deep Reinforcement Learning Path Planning for Aerial Wildfire Monitoring in Partially Observable Environments. In *2024 American Control Conference (ACC)*, pages 601–607, 2024.

[14] Maxime Collignon, Adolfo Perrusquía, Antonios Tsourdos, and Weisi Guo. Search and rescue operations in wildfires using unmanned aerial vehicles: A multi-agent deep reinforcement learning approach. *Neurocomputing*, 653:131211, 2025.

[15] Hamza Chakraa and Murat Bronz. Accelerating Wildfire Spread Prediction for Guidance: A ConvLSTM Deep Learning Approach. In *AIAA SCITECH 2026 Forum*, Orlando, FL, 2026.

[16] John Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[17] Edouard Rolland, Kilian Meier, Murat Bronz, Aditya Shrikhande, Tom Richardson, Ulrik Pagh Schultz Lundquist, and Anders Christensen. WildBridge: Ground station interface for lightweight multi-drone control and telemetry on dji platforms. *Proceedings of the 13th International Conference on Robot Intelligence Technology and Applications (RiTA 2025)*, 2025.

[18] Hamza Chakraa, Matthieu Verdoucq, João Machado, and Murat Bronz. A preliminary study on smoke plume observation with drone swarms. In *14th International Conference on Unmanned Aircraft Systems (ICUAS)*, Corfu, Greece, 2026.

http://www.imavs.org/

Distributed Formation Control based on Bounded Voronoi Tessellation and Cable Tension for Cooperative Hovering of a Cable-Suspended Load

Saleh Abuali[1,#], Natalio Mendoza-Robles[1,2,#],
Bogdan Robu[1], Nicolas Marchand[1], Sylvain Durand[2], and Lara Briñón-Arranz[1]
[#]These authors contributed equally and are considered joint first authors.
[1]Univ. Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab, 38000 Grenoble, France *
[2]Univ. Strasbourg, CNRS, INSA Strasbourg, ICube, 67000 Strasbourg, France †

http://www.imavs.org/

ABSTRACT

This paper addresses the distributed formation control of N Unmanned Aerial Vehicles (UAVs) cooperatively carrying a cable-suspended payload. This is achieved using only local information, without explicit inter-agent communication, an assignment of UAVs to predefined positions in the formation, or knowledge of the load's position. The key observation is that, when hovering at the same altitude, the horizontal component of the cable tension pulls each UAV towards the position of the load's planar projection, acting as a built-in cohesion term and leaving the controller to supply only separation. We realise this separation using a bounded Voronoi tessellation-based method and derive the gain that places the team at any desired formation radius in closed form. An equilibrium analysis predicts the radius from the balance between the separation force and the cable tension. This approach has been validated through simulations and real-world flights involving four mini-UAVs.



Figure 1: Experimental setup: four Parrot MAMBO mini-UAVs cooperatively carrying a cable-suspended load. A video of the experiments is available at `youtu.be/fomDca2OPEA`.

1 INTRODUCTION

Unmanned Aerial Vehicles (UAVs) are increasingly deployed for aerial transportation, from package and medical supplies delivery to the handling of construction materials [1]. A single UAV, however, is limited by its own payload capacity; deploying several UAVs to carry a common load overcomes this limitation and yields a modular transportation system [2].

Cooperative aerial transport has been approached along a spectrum of coordination paradigms. Centralised schemes compute the team's commands from full knowledge of the system state, achieving precise load control at the cost of a single point of failure and limited scalability [3, 4]. Decentralised schemes distribute the computation across the team, typically relying on inter-agent communication to maintain a predefined geometric formation [5, 6, 7]. Fully decentralised, communication-free approaches remove this dependency, but it typically relies on tension-based control, which is sensitive to tension estimation errors [8, 9].

A recurring requirement across these methods is that the formation geometry is fixed in advance, either through a reference assigned to each agent or through prescribed inter-agent distances over a rigid graph [10]. Such a predefined assignment couples each UAV to specific partners and must be designed and certified before deployment. This motivates a complementary question: can a regular formation emerge from local interactions alone, without assigning UAVs to formation slots and without communicating absolute or relative references?

This question is naturally framed in terms of flocking. The control of a UAV swarm from local sensing is classically posed as the combination of three behaviours [11]: cohesion, separation, and alignment, of which cohesion and separation are the two that shape the formation geometry. Cohe-

*Email: firstname.lastname@gipsa-lab.grenoble-inp.fr
†Email: sdurand@unistra.fr

http://www.imavs.org/

sion draws the agents together, while separation spreads them apart; their balance sets the spacing of the team. The key observation of this work is that, in a formation of UAVs with a cable-suspended load, cohesion need not be generated by the controller at all. The horizontal component of the cable tension pulls every UAV toward the common load point, acting as a built-in cohesion term that requires neither measurement nor communication. The controller is then left to supply only separation behavior.

For the separation behaviour, we draw on the coverage literature, where teams of agents are distributed over a region by spreading out to maximise sensing area while avoiding overlaps and blind spots. This is frequently addressed with spatial partitioning methods such as Voronoi tessellations [12], in which each agent is responsible for the region around its own position, and which operates under restricted sensing ranges to ensure decentralised, collision-free coordination [13, 14, 15]. We adopt a Bounded Voronoi Tessellation (BVT) introduced in [15] as the separation mechanism: it enforces inter-agent avoidance using only local relative positions, without online optimisation and without a predefined configuration. Combined with the tension-supplied cohesion, it yields an evenly spaced circular formation that emerges from local sensing, without communication, without predefined formation position assignments, and without knowledge of the load position.

The contributions of this paper are threefold. First, we identify and formalise, through an equilibrium analysis of the coupled UAV-load system, the horizontal cable tension as a built-in cohesion term. Second, we design a bounded-Voronoi separation law with a closed-form gain that places the formation at any desired radius, and we analyse the radial stability of the resulting equilibrium. Third, we validate the approach in simulation and in real flights with a team of Parrot MAMBO mini-UAVs.

2 PROBLEM FORMULATION

Consider a group of N identical UAVs of mass m cooperatively carrying a point-mass payload of mass m_L. UAV i is located at $\boldsymbol{p}_i \in \mathbb{R}^3$ and connected to the payload at $\boldsymbol{p}_L \in \mathbb{R}^3$ by a taut, massless cable of fixed length $l > 0$. Let $\boldsymbol{\xi}_i = \frac{\boldsymbol{p}_L - \boldsymbol{p}_i}{\|\boldsymbol{p}_L - \boldsymbol{p}_i\|}$ be the unit vector pointing from UAV i toward the payload for each $i \in \{1, \dots, N\}$. The taut cable exerts on UAV i a tension force $\boldsymbol{T}_i = \mathcal{T}_i \boldsymbol{\xi}_i$ of magnitude $\mathcal{T}_i = \|\boldsymbol{T}_i\| \geq 0$, pulling it toward the load; by reaction, the cable pulls the load toward UAV i. Each UAV carries a local sensor that estimates the relative position of neighbours within a sensing radius R (see Figure 2).

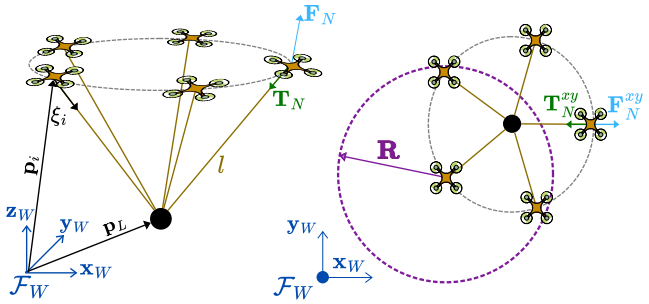


Figure 2: Overview of the cooperative load carrying system. Left: N UAVs cooperatively hover while suspending a point-mass load via taut cables of length l, where unit vectors $\boldsymbol{\xi}_i$ and tensions $\boldsymbol{T}_i$ are shown in the world frame $\mathcal{F}_W$. Right: each UAV senses the relative positions of neighbours within its sensing radius R.

2.1 System Dynamics

The simplified translational dynamics of UAV i and the payload are [9]:

$$m\,\dot{\boldsymbol{v}}_i = \boldsymbol{F}_i - mg\,\boldsymbol{z}_W + \boldsymbol{T}_i, \tag{1}$$

$$m_L\,\ddot{\boldsymbol{p}}_L = \boldsymbol{T}_{\text{res}} - m_L g\,\boldsymbol{z}_W, \qquad \boldsymbol{T}_{\text{res}} := \sum_{i=1}^{N} -\mathcal{T}_i\,\boldsymbol{\xi}_i, \tag{2}$$

where $\boldsymbol{v}_i = \dot{\boldsymbol{p}}_i$, $\boldsymbol{F}_i \in \mathbb{R}^3$ is the combined motor thrust expressed in the world frame $\mathcal{F}_W = [\boldsymbol{x}_W, \boldsymbol{y}_W, \boldsymbol{z}_W]$, where $\boldsymbol{z}_W = [0, 0, 1]^\top$ is the world vertical axis (opposing gravity), and g is gravitational acceleration. The load is driven by the resultant cable force $\boldsymbol{T}_{\text{res}}$, since each cable pulls it toward the corresponding UAV (along the direction $-\boldsymbol{\xi}_i$). We assume a low-level attitude control loop such that the thrust direction tracks its reference sufficiently fast.

2.2 Interaction Graph

The interaction of the team is described by a graph $\mathcal{G} := (\mathcal{U}, \mathcal{E})$, where $\mathcal{U} = \{1, \dots, N\}$ is the set of vertices representing the robots (UAVs) and $\mathcal{E} \subset \{(i, j) \in \mathcal{U} \times \mathcal{U} : i \neq j\}$ the set of edges (sensing links). $\mathcal{G}$ is *connected* if a path joins any two vertices. The neighbour set of robot i within a sensing radius R is defined by $\mathcal{N}_i^R := \{\, j \in \mathcal{U} : (i, j) \in \mathcal{E} \,\}$. We assume the sensing radius R is large enough that $\mathcal{G}$ remains connected throughout the mission .

2.3 Control Objective

With a view to equally distribute the load among the N UAVs, our objective is to stabilise the group of UAVs to an evenly spaced planar circular formation at a desired altitude z^{des} in a distributed manner, using only limited-range local sensing: without a predefined position assignment, and without knowledge of the load position $\boldsymbol{p}_L$. This objective concerns only the geometry of the formation and involves no global reference: the absolute positions of the group and of

http://www.imavs.org/

the load are left unconstrained, so the team may drift as a whole while the formation converges. Steering the load along a desired trajectory under the same sensing constraints is beyond the scope of this paper.

The analysis and the control design rely on the following assumptions: (A1) the cables are massless, inextensible, of identical known length l, attached to the load at a common point, and remain taut throughout the flight; (A2) all UAVs regulate the same altitude z^{des}, so the formation evolves in a horizontal plane above the load; (A3) the UAVs are identical, and m, m_L, l, N and g are known to each of them; (A4) each UAV measures the relative positions of its neighbours within the sensing radius R, together with its own position, velocity and altitude, with no inter-agent communication and no measurement of $\boldsymbol{p}_L$ or of the tensions $\boldsymbol{T}_i$; (A5) R is large enough for the sensing graph $\mathcal{G}$ to remain connected throughout the mission; (A6) a low-level attitude controller tracks the commanded thrust direction sufficiently fast, so the thrust vector $\boldsymbol{F}_i$ is treated as the control input.

3 Control Strategy

As introduced in Section 1, cohesion and separation are the two flocking behaviours that shape the formation geometry. The key observation of this work is that, in cooperative hovering with a cable-suspended load, cohesion is supplied by physics rather than by control. For a team hovering at a desired altitude z^{des}, the X–Y projection of the tension acting on UAV i is

$$\boldsymbol{T}_i^{xy} = \mathcal{T}_i \frac{\boldsymbol{p}_L^{xy} - \boldsymbol{p}_i^{xy}}{\|\boldsymbol{p}_L - \boldsymbol{p}_i\|}, \tag{3}$$

i.e., a force directed from $\boldsymbol{p}_i^{xy}$ toward the $X-Y$ load position $\boldsymbol{p}_L^{xy}$, where the $(\cdot)^{xy}$ superscript represents the X and Y components of the vector. Since all cables meet at the common load point and remain taut under the load's weight (hence $\|\boldsymbol{p}_L - \boldsymbol{p}_i\| = l$), every UAV is pulled toward a single shared point: the formation centre. Structurally, (3) is a consensus term on the horizontal positions, realised for free by the physical coupling rather than by inter-agent communication. Cohesion is thus inherent to the system, and the controller needs only supply *separation*, for which we propose the Bounded Voronoi Tessellation (BVT) method.

3.1 Separation Mechanism based on Voronoi Tessellation

In the BVT, each Voronoi region is enclosed within a 'bubble' (Figure 3). First, the method establishes the set of detected neighbours $\mathcal{N}_i^R$, within a sensing radius R centered at robot i. A Voronoi tessellation is then computed from the relative positions of the detected neighbours, partitioning the plane according to proximity. The Voronoi region of robot i is defined as follows:

$$V_i = \{\mathbf{q} \in \mathbb{R}^2 : \|\mathbf{q} - \mathbf{p}_i\| \leq \|\mathbf{q} - \mathbf{p}_j\|, \forall j \in \mathcal{N}_i^R\}, \tag{4}$$

i.e., it represents the set of points closer to $\mathbf{p}_i$ than to any other detected neighbour. We define a bubble centered on the

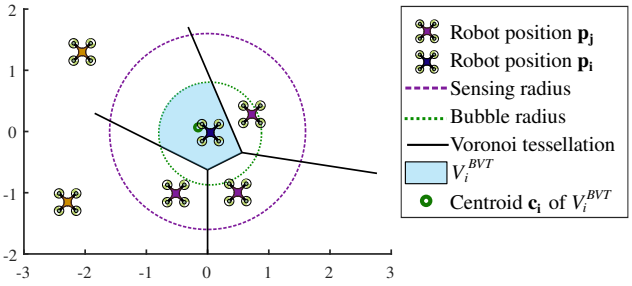


Figure 3: Bubble-based Voronoi tessellation.

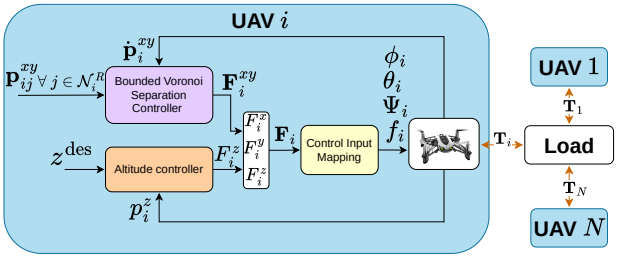


Figure 4: Control architecture. For each UAV i, the separation controller computes the horizontal force $\boldsymbol{F}_i^{xy}$ from the relative neighbour positions within the sensing radius R, and a PID altitude controller regulates the altitude via F_i^z. The resultant thrust $\boldsymbol{F}_i$ is then mapped to the UAV control inputs (tilt angles and thrust magnitude).

position of robot i, such that

$$\mathcal{B}_i = \{\mathbf{q} \in \mathbb{R}^2 : \|\mathbf{q} - \mathbf{p}_i\| \leq r_v\}, \tag{5}$$

where $r_v > 0$ is the bubble radius, which serves as the boundary for the Voronoi cell. Consequently, the safe bounded Voronoi region of each robot is the intersection of its Voronoi region with its bubble, such that

$$V_i^{BVT} = V_i \cap \mathcal{B}_i, \tag{6}$$

and the centroid of each cell V_i^{BVT} is defined as

$$\mathbf{c}_i = \frac{\int_{V_i^{BVT}} \mathbf{q}\,\rho(\mathbf{q})\,d\mathbf{q}}{\int_{V_i^{BVT}} \rho(\mathbf{q})\,d\mathbf{q}}, \tag{7}$$

where $\rho(\cdot)$ is a density function, set to a uniform $\rho \equiv 1$ in this study [16, 14].

Since the centroid represents the most isolated point within V_i^{BVT} with respect to its neighboring cells, driving UAV toward its cell centroid, $\mathbf{p}_i^{xy} \to \mathbf{c}_i$, naturally promotes separation among agents. The corresponding control law used to achieved this behavior is presented in the following section.

3.2 Control Input Design

As stated in Section 2.1, we assume direct control over the thrust vector in $\mathcal{F}_W$. The overall control architecture is shown

in Figure 4. First, we regulate the altitude of each UAV with a standard PID controller acting on the altitude error, which sets the Z component of the thrust F_i^z. Then, we compute the horizontal thrust $\boldsymbol{F}_i^{xy}$ with a PD-like law that draws the UAV toward its Voronoi centroid $\boldsymbol{c}_i$, using the BVT approach as follows:

$$\boldsymbol{F}_i^{xy} = m\left(K_s(\mathbf{c}_i - \mathbf{p}_i^{xy}) - K_d\,\dot{\mathbf{p}}_i^{xy}\right), \tag{8}$$

where $\mathbf{c}_i$ is the centroid of the UAV's Voronoi cell computed in (7), and K_s, K_d are the separation and velocity-damping control gains, respectively.

Since the separation force and the horizontal tension oppose each other, the UAV does not converge to the centroid $\mathbf{c}_i$ but at an equilibrium where the two forces balance, i.e., $\|\boldsymbol{F}_i^{xy}\| = \|\boldsymbol{T}_i^{xy}\|$. We analyse this equilibrium and design K_s to achieve a desired formation radius in the following section.

4 Equilibrium Force Analysis and Formation Shaping

The equilibrium formation arises from the radial balance, in the X–Y plane, between the projection of the thrust and the projection of the cable tension (see Figure 5). Since the cables of identical length l are assumed to stay taut, every UAV is equidistant from the load, fixing a common formation radius r_c about the origin $O = \boldsymbol{p}_L^{xy}$. Assuming that the team is initialised close to a symmetric circular configuration around the load, the separation mechanism in turn spreads the UAVs evenly, yielding a uniform circular formation. The tension is fixed by the physical cable parameters, the load mass, and the number of agents; hence, for a desired formation radius $r_c^\star$, the design goal is to choose a separation gain K_s that yields $r_c = r_c^\star$.

4.1 Cable Tension in Uniform Circular Formation

At an equilibrium uniform circular formation, each UAV carries a share $m_L g/N$ of the payload weight, and the horizontal component of the cable tension is [9]

$$\|\boldsymbol{T}_i^{xy}(r_c)\| \;=\; \frac{m_L g}{N}\,\frac{r_c}{\sqrt{l^2 - r_c^2}}, \tag{9}$$

which pulls the UAV towards O and diverges as $r_c \to l$, enforcing the cable-length constraint.

4.2 Equilibrium Separation Force

For a stationary symmetric circular formation with $\dot{\mathbf{p}}_i^{xy} = \mathbf{0}$, the damping term in (8) drops out and the separation force reduces to

$$\mathbf{F}_i^{xy} = m\,K_s(\mathbf{c}_i - \mathbf{p}_i^{xy}), \tag{10}$$

directed radially outward. By symmetry, the centroid $\mathbf{c}_i$ lies on the radial axis through $\mathbf{p}_i^{xy}$; let $c_i(r_c) := \|\mathbf{c}_i - O\|$ denote its distance from the load centre O. The separation force then has magnitude $mK_s\,\sigma(r_c)$, where

$$\sigma(r_c) \;=\; c_i(r_c) - r_c \;>\; 0 \tag{11}$$

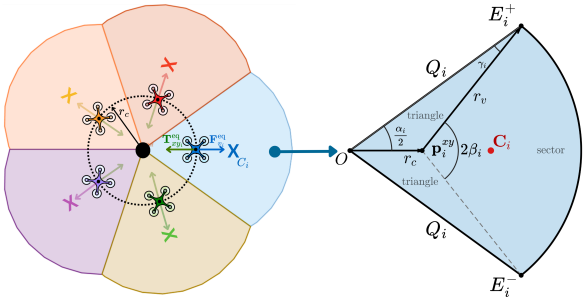


Figure 5: Decomposition of the Voronoi cells at equilibrium.

is the radial offset between UAV i and the centroid of its bounded Voronoi cell.

We evaluate $c_i(r_c)$ in the operating regime $r_v > r_c$, i.e., the bubble radius exceeds the formation radius. As described in Fig. 5, at the symmetric equilibrium, each pair of adjacent neighbours is equidistant from O, so their bisectors meet exactly at O. Each Voronoi cell is then bounded by two straight edges originating at O, with angle $\alpha_i = 2\pi/N$ and by the disk arc of radius r_v. Let $E_i^\pm$ be the points where the bisectors exit the disk. The equilibrium Voronoi cell $V_i^{BVT\star}$ admits an exact decomposition into *two triangles* $\triangle O\,\boldsymbol{p}_i^{xy}\,E_i^+$ and $\triangle O\,\boldsymbol{p}_i^{xy}\,E_i^-$, plus a *circular sector* $\mathcal{S}(\boldsymbol{p}_i^{xy},\, r_v,\, 2\beta_i)$ of radius r_v centred at $\boldsymbol{p}_i^{xy}$ and spanning the angle 2β:

$$V_i^{BVT\star} \;=\; \triangle O\boldsymbol{p}_i^{xy}E_i^+ \,\cup\, \triangle O\boldsymbol{p}_i^{xy}E_i^- \,\cup\, \mathcal{S}(\boldsymbol{p}_i^{xy},\, r_v,\, 2\beta_i). \tag{12}$$

To obtain the area of the cell, we compute the angle γ (see Figure 5) by applying the law of sines in the triangle $O\boldsymbol{p}_i^{xy}E_i^+$, with the angle $\alpha_i/2$ at O opposite the side r_v and angle γ_i opposite the side r_c. The sector half-angle β_i and the triangle side $Q_i = |OE_i^\pm|$ are then

$$\gamma_i = \arcsin\!\left(\frac{r_c\sin(\pi/N)}{r_v}\right), \tag{13}$$

$$\beta_i = \frac{\pi}{N} + \gamma_i, \tag{14}$$

$$Q_i = r_c\cos\frac{\pi}{N} + \sqrt{r_v^2 - r_c^2\sin^2\frac{\pi}{N}}. \tag{15}$$

The two triangles and the circular sector have the following area and centroid distance from O:

piece	area	centroid from O
triangle	$r_c Q_i \sin(\pi/N)$	$\dfrac{r_c + Q_i\cos(\pi/N)}{3}$
sector	$r_v^2\,\beta_i$	$r_c + \dfrac{2r_v\sin\beta_i}{3\beta_i}$

(16)

The area-weighted centroid distance from O is therefore

$$c_i(r_c) \;=\; \frac{r_c Q_i \sin\frac{\pi}{N}\cdot\frac{r_c+Q_i\cos(\pi/N)}{3} \;+\; r_v^2\beta\left(r_c + \frac{2r_v\sin\beta_i}{3\beta_i}\right)}{r_c Q_i\sin\frac{\pi}{N} + r_v^2\beta_i} \tag{17}$$

http://www.imavs.org/

which, together with (11), fixes the equilibrium offset $\sigma(r_c)$ entering the control law.

4.3 Equilibrium Condition and Gain Design

The steady-state radial balance equates the outward separation force to the inward tension at $\dot{\mathbf{v}}_i = \mathbf{0}$:

$$\begin{aligned} \|\boldsymbol{F}_i^{xy}\| &= \|\boldsymbol{T}_i^{xy}(r_c)\| \\ m\,K_s\,\sigma(r_c) &= \frac{m_L g}{N}\,\frac{r_c}{\sqrt{l^2 - r_c^2}}. \end{aligned} \tag{18}$$

To achieve a desired circular formation of radius $r_c^\star$, we solve (18) for the separation gain,

$$K_s = \frac{m_L g}{mN}\,\frac{r_c^\star}{\sqrt{l^2 - (r_c^\star)^2}}\,\frac{1}{\sigma(r_c^\star)}. \tag{19}$$

The balance (18) admits a single equilibrium radius. As r_c grows, the tension term (9) increases monotonically and diverges at $r_c = l$, while the separation offset $\sigma(r_c)$ decreases as the cell spreads wider, and its centroid is drawn inward. The two curves therefore cross exactly once, and the crossing is stable: an outward perturbation raises the tension and lowers the separation, restoring the UAV inward, and vice versa, so the formation self-regulates to a single radius without explicit radius feedback.

4.4 Stability of the Equilibrium Formation

The equilibrium has two independent modes: the formation *radius* and the *angular* spacing of the UAVs. This work focuses on the radial mode, which the gain design (19) directly shapes; the angular spacing is a consequence of the separation mechanism, as discussed below.

At fixed angular positions, each UAV moves radially under the balance between the outward separation force $mK_s\,\sigma(r_c)$ and the inward horizontal tension $\|\boldsymbol{T}_i^{xy}(r_c)\|$. As r_c increases, the tension (9) grows steadily and diverges at $r_c = l$, while the separation offset $\sigma(r_c)$ decreases as the Voronoi cell spreads wider and its centroid is drawn inward. The two terms therefore intersect at a single radius, the designed $r_c^\star$ of (19), and this intersection is stable: an outward perturbation raises the tension and lowers the separation, pulling the UAV back inward, while an inward perturbation does the reverse. The velocity damping term $-mK_d\dot{\boldsymbol{p}}_i^{xy}$ in (8) damps the resulting radial motion, so the formation self-regulates to $r_c^\star$ without explicit radius feedback.

The angular spacing is handled by the same separation mechanism: when $\alpha_i \neq 2\pi/N$, the Voronoi cell centroid $\mathbf{c}_i$ has a tangential component that drives the UAV toward the wider gap, so equal angular spacing $\alpha_i = 2\pi/N$ is the configuration in which this component vanishes. The formation orientation about $\boldsymbol{z}_W$ remains free, as no UAV measures an absolute angle. Accordingly, the experimental analysis that follows evaluates the radial behaviour through the formation radius $\hat{r}_c$, and does not separately characterise the angular spacing.

5 SIMULATION AND EXPERIMENTAL RESULTS

This section presents both the simulation and the experimental results of our Bounded Voronoi-based formation control approach.

As mentioned in Section 4, to start the controller near equilibrium and avoid a local minimum where one UAV ends up at the centre of the formation [15] (which slackens its cable), the UAVs first take off and hover on a predefined circular formation of radius 0.6 m above the load using a standard PID position controller. The position controller is then switched off and replaced by the proposed Voronoi-based separation controller (8), whose gain K_s is computed from (19) for the desired radius $r_c^\star$. No position reference is given after the switch: the formation expands or contracts purely under the balance between the separation force and the cable tension, and it is stabilised at the desired equilibrium radius.

5.1 Control Input Mapping

The designed control input throughout the paper has been the thrust vector $\boldsymbol{F}_i$. For our mini-UAVs, however, the actual control inputs to each UAV are the tilt angles (roll ϕ_i, pitch θ_i, and yaw ψ_i) together with the thrust magnitude f_i. Since yaw can be managed separately, we assume without loss of generality that $\psi_i = 0$ for all UAVs. The control input $\mathbf{U}_i = [\phi_i, \theta_i, \psi_i, f_i]^\top$ is then retrieved by decomposing $\boldsymbol{F}_i$ into a thrust magnitude and tilt angles. Adopting the Z–Y–X Euler convention, in which UAV i executes the rotations ψ_i, θ_i, ϕ_i in turn, gives

$$\phi_i = \operatorname{atan2}\left(F_i^y,\ \sqrt{F_i^{z\,2} + F_i^{x\,2}}\right) \tag{20}$$

$$\theta_i = \operatorname{atan2}(F_i^x,\ F_i^z) \tag{21}$$

$$f_i = \|\boldsymbol{F}_i\|, \tag{22}$$

where $\operatorname{atan2}$ is the two-argument arc-tangent function.

5.2 Formation Radius Estimation

In real experiments, the UAVs may converge to uneven angular spacing around the circle; we therefore estimate the formation radius as the mean radial distance to the load centre, which is insensitive to how the UAVs are distributed around it. As established in Section 4, the relevant radius is the offset from the load centre $O = \mathbf{p}_L^{xy}$, so we use the load position provided by the motion-capture system (only for evaluation, not in the control loop) and compute the individual radii $r_i(t)$ for each UAV

$$r_i(t) = \|\mathbf{p}_i^{xy}(t) - \mathbf{p}_L^{xy}(t)\|, \tag{23}$$

and the estimated circular formation radius

$$\hat{r}_c(t) = \frac{1}{N}\sum_{i=1}^{N} \left\|\mathbf{p}_i^{xy}(t) - \mathbf{p}_L^{xy}(t)\right\|. \tag{24}$$

The steady-state accuracy is then quantified by the root-mean-square (RMS) error between the estimated and the de-

http://www.imavs.org/

sired radius,

$$e_{\mathrm{RMS}} = \sqrt{\frac{1}{M} \sum_{t_k \in [t_1,\, t_2]} \left(\hat{r}_c(t_k) - r_c^\star\right)^2}, \qquad (25)$$

where $[t_1,\, t_2]$ is a steady-state window and M is the number of motion-capture samples it contains.

5.3 Simulation results

To test the proposed control approach in simulation, we use a Matlab/Simulink simulator that includes a realistic model of the Parrot MAMBO mini-UAVs, with the full nonlinear dynamics and the experimentally identified closed-loop angle and thrust dynamics from [9]. The cables are modelled as rigid, massless bars that transmit force between each UAV and the load.

To validate the closed-form gain design (19), we simulate the system for several desired radii $r_c^\star \in \{0.5,\, 0.6,\, 0.7,\, 0.8,\, 0.9,\, 1.0\}$ m; from (19), the corresponding separation gain for each case is recomputed as $K_s \in \{0.4835,\, 0.6261,\, 0.7961,\, 1.0037,\, 1.2639,\, 1.6018\}$. The simulated team consists of $N = 5$ UAVs of mass $m = 78.5$ g cooperatively carrying a point-mass load of $m_L = 50$ g through identical cables of length $l = 1.5$ m. The bubble radius is set to $r_v = 1.8$ m and the sensing radius to $R = 2.8$ m.

Figure 6 shows the time evolution of the estimated formation radius $\hat{r}_c$ (24) for each desired radius. During the first phase ($t = 0$–10 s) the PID position controller holds the team on the 0.6 m hovering circle; at the switch the formation expands or contracts purely under the imbalance between the separation force and the cable tension, and settles at the designed radius. The realised steady-state radius exactly matches the design value across all tests, confirming that the equilibrium analysis of Section 4 correctly predicts the closed-loop geometry.

5.4 Real-World Experiments

For the real-world experiments, we use a set of Parrot MAMBO mini-UAVs in a flight arena equipped with a motion-capture (MoCap) system. The decentralised control strategy is emulated within a centralised Matlab/Simulink framework. While the MoCap system provides global position measurements for all UAVs, local sensing is reproduced by constructing an individual neighbor set for each agent. Specifically, for UAV i, only agents whose distance to UAV i is less than or equal to the sensing radius R are included in its neighborhood and used in the control-law evaluation. Each UAV also has access to its own position and velocity estimates. The computed control inputs are then transmitted to each UAV over a Bluetooth link.

The experimental platform consists of $N = 4$ Parrot MAMBO mini-UAVs attached to a common point-mass load of $m_L = 33.5$ g by cables of identical length $l = 1.4$ m (see Figure 1). The sensing radius is set to $R = 3$ m, the bubble

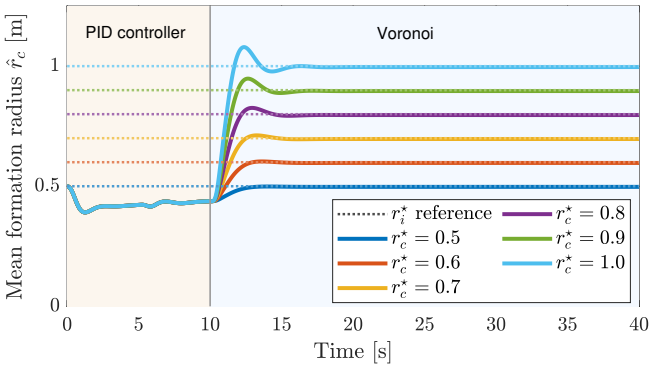


Figure 6: Simulation with $N = 5$ UAVs: estimated formation radius $\hat{r}_c$ (24) for six desired radii $r_c^\star \in \{0.5, \ldots, 1.0\}$ m (solid), with the corresponding references (dotted). The shaded regions mark the PID hovering phase and the proposed Voronoi-based controller.

radius to $r_v = 1.5$ m, and the separation gain $K_s = 1.0423$ computed from (19) for the desired radius $r_c^\star = 0.7$ m.

The experiment is divided into three phases, marked by the shaded regions in Figure 7. First (0–7 s), the four UAVs take off freely from the ground. Second (7–15 s), they lift the load and hover on a circle of radius 0.6 m at the desired altitude using a PID position controller; the cables become taut and the load is airborne by the end of this phase. Once the load is airborne, the horizontal tension component (3) acts on each UAV as an inward disturbance that the position controller does not compensate perfectly, so the realised hovering radius settles below the commanded 0.6 m circle (visible in Figure 7 before $t = 15$ s); the same effect appears in the simulation of Figure 6. This residual offset is a manifestation of the tension-supplied cohesion exploited by the proposed approach. Third (from $t = 15$ s), the position controller is switched off and replaced by the Voronoi-based separation controller (8), and the formation is left to self-regulate with no position reference and no knowledge of the load position. Since the designed equilibrium radius exceeds the hovering one, the switch produces a visible expansion of the formation, driven only by the imbalance between the centroid offset and the cable tension.

5.5 Analysis of the Results

Figure 7 shows the time evolution of the estimated radius $\hat{r}_c$ from (24) together with the individual radii r_i from (23). After the switch at $t = 15$ s, the formation expands from the hovering configuration and converges to the designed radius with no position reference, settling at a mean of $\hat{r}_c = 0.690$ m over the window $t = 22$–32 s, with an RMS error of 2.0 cm with respect to $r_c^\star = 0.7$ m, i.e. 2.8% of the desired radius, reached purely through the balance between the cable tension and the Voronoi-based separation. The mean radius therefore matches the design value closely, while the individual

http://www.imavs.org/

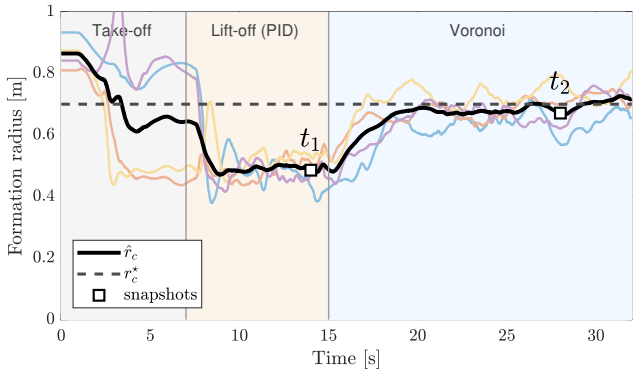


Figure 7: Experiment with 4 Parrot MAMBO mini-UAVs for $r_c^\star = 0.7$ m: estimated formation radius $\hat{r}_c$ (24) (solid black), individual radii r_i (thin), and desired radius (dashed). The shaded regions mark the three phases: free take-off, lift-off with the load under the PID position controller, and the proposed Voronoi-based controller. The square markers indicate the snapshot instants t_1 and t_2 of Figure 8.

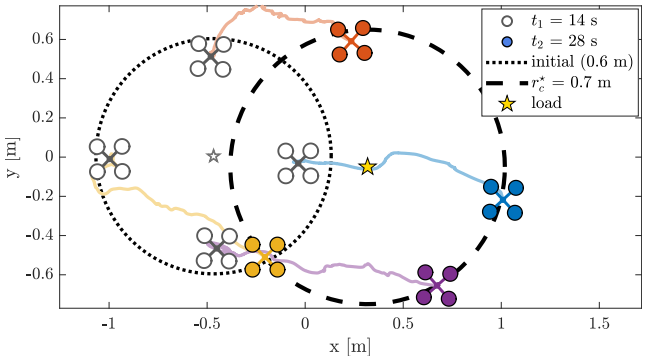


Figure 8: Top view of the experiment with 4 Parrot MAMBO UAVs at two instants: $t_1 = 14$ s, the predefined 0.6 m hovering circle under the PID controller (just before the switch), and $t_2 = 28$ s, the expanded formation under the Voronoi-based controller, near the designed radius $r_c^\star = 0.7$ m. The star marks the load centre O.

radii differ from one another: under taut cables at a common altitude, $r_i = \sqrt{l^2 - h_i^2}$ with h_i the height of UAV i above the load, so the imperfect altitude tracking of each UAV causes a radial offset. The spread of the individual radii r_i around $\hat{r}_c$ also reflects the angle biases of the UAVs, whose realised tilt angles differ from the commanded ones, $\theta^{\text{real}} = \theta^{\text{command}} + \theta^{\text{bias}}$, together with hardware differences such as battery health and propeller wear.

Figure 8 gives the corresponding top view through two formation snapshots. At $t_1 = 14$ s, just before the switch, the UAVs maintain the predefined 0.6 m circle under the PID position controller; at $t_2 = 28$ s, after convergence of the Voronoi-based controller, the formation has expanded to the designed radius $r_c^\star = 0.7$ m, driven only by the tension–separation balance. The UAVs do not form a perfectly uniform circle due to the differing biases of each UAV.

6 CONCLUSIONS

This paper presents a distributed strategy for cooperative hovering cable-suspended payload by N UAVs, in which the team self-organises into an evenly spaced circular formation using only local sensing. The key finding is that the horizontal component of the cable tension acts as a built-in cohesion term, meaning that the controller only needs to supply separation. This was achieved using a bounded Voronoi-based controller with a closed-form gain that positions the formation at any desired radius.

This approach was validated through simulations and real-world flights involving four Parrot MAMBO mini-UAVs. In these experiments, the team converged to the designed radius with an RMS error of 2.8%, without any position reference, predefined assignment or knowledge of the load position. The main limitations are twofold. First, the controller contains no global objective, so the absolute position of the group is unconstrained and the team drifts as a whole. Second, the gain design in (19) requires knowledge of the load mass m_L and the number of agents N. Building upon the payload-carrying framework presented in this work, future research will focus on analysing the stability of the formation during motion and extending the approach to the payload transportation problem. Furthermore, distributed estimation techniques will be developed to infer the load mass and the number of agents online using only local measurements.

ACKNOWLEDGEMENTS

This work has been partially supported by the French National Research Agency in the context of the muteSWARM project (ANR-23-IAS2-0004) and the French National Research Infrastructure ROBOTEX 2.0 (Grants ROBOTEX ANR-10-EQPX-44-01 and TIRREX ANR-21-ESRE-0015).

REFERENCES

[1] Ladfah Gantiya and Chuchart Pintaviooj. Medical delivery drone. In *2024 16th Biomedical Engineering International Conference (BMEiCON)*, pages 1–6, 2024.

[2] Taeyoung Lee. Geometric control of quadrotor UAVs transporting a cable-suspended rigid body. *IEEE Transactions on Control Systems Technology*, 26(1):255–264, 2018.

[3] Taeyoung Lee, Koushil Sreenath, and Vijay Kumar. Geometric control of cooperating multiple quadrotor UAVs with a suspended payload. In *52nd IEEE Conference on Decision and Control*, pages 5510–5515, 2013.

http://www.imavs.org/

[4] Sihao Sun, Xuerui Wang, Dario Sanalitro, Antonio Franchi, Marco Tognon, and Javier Alonso-Mora. Agile and cooperative aerial manipulation of a cable-suspended load. *Science Robotics*, 10(107), 2025.

[5] Xiaozhen Zhang, Fan Zhang, Panfeng Huang, Jiale Gao, Hang Yu, Chongxu Pei, and Yizhai Zhang. Self-triggered based coordinate control with low communication for tethered multi-UAV collaborative transportation. *IEEE Robotics and Automation Letters*, 6(2):1559–1566, 2021.

[6] Nicola De Carli, Riccardo Belletti, Emanuele Buzzurro, Andrea Testa, Giuseppe Notarstefano, and Marco Tognon. Distributed nmpc for cooperative aerial manipulation of cable-suspended loads. *IEEE Robotics and Automation Letters*, 10(10):10546–10553, 2025.

[7] Natalio Mendoza-Robles, Alfredo Namigtle-Jiménez, Pierre Susbielle, Nicolas Marchand, Sylvain Durand, Rogerio Enríquez-Caldera, Lara Briñón-Arranz, and José Fermi Guerrero-Castellanos. Cooperative control of multirotor UAVs with disturbance rejection capabilities for load transportation. *IEEE Access*, 14:67204–67217, 2026.

[8] C. Gabellieri, M. Tognon, D. Sanalitro, and A. Franchi. Force-based pose regulation of a cable-suspended load using UAVs with force bias. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6920–6926, 2023.

[9] Saleh Abuali, Kostas Skantzikas, Pierre Susbielle, Nicolas Marchand, and Lara Briñón-Arranz. Communication-free cooperative transportation with multiple UAVs. *IFAC-PapersOnLine*, 59(18):187–192, 2025.

[10] Héctor García de Marina and Ewoud Smeur. Flexible collaborative transportation by a team of rotorcraft. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1074–1080, 2019.

[11] Reza Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3):401–420, 2006.

[12] J. Cortés, S. Martínez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, 2004.

[13] Meng Zhou, Jianyu Li, Chang Wang, Jing Wang, and Li Wang. Applications of Voronoi Diagrams in Multi-Robot Coverage: A Review. *Journal of Marine Science and Engineering*, 12(6):1–21, 2024.

[14] Federico Pratissoli, Beatrice Capelli, and Lorenzo Sabattini. On Coverage Control for Limited Range Multi-Robot Systems. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9957–9963. IEEE, 2022.

[15] Natalio Mendoza-Robles, Lara Briñón-Arranz, Pierre Susbielle, Kostas Skantzikas, Sylvain Durand, and Nicolas Marchand. Distributed formation control with local sensing combining Bubble-based Voronoi tessellation and consensus. In *2026 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 805–812, 2026.

[16] Vladimir Alexandrov, Konstantin Kirik, and Alexander Kobrin. Multi-robot Voronoi tessellation based area partitioning algorithm study. *Paladyn, Journal of Behavioral Robotics*, 9:214–220, 2018.

http://www.imavs.org/

A Structure-Aware Reactive Motion Planner for Autonomous Nano-UAV Navigation in Confined 3D Tunnels Using Minimal Range Sensing

Abner Asignacion Jr*, and Satoshi Suzuki
Chiba University, Japan

http://www.imavs.org/

Abstract

Inspection of confined tunnel environments remains challenging for small robotic platforms. While snake and soft robots can operate in highly constrained spaces, their effectiveness degrades in long, elevated, and vertically structured tunnels. Micro- and nano-UAVs offer a compelling alternative, but their deployment is severely constrained by limited sensing and onboard computation, rendering map-based planning impractical. This paper presents an autonomous structure-aware motion planning (SAMP) strategy for nano-UAV navigation in narrow tunnels using only sparse range sensing. The proposed method explicitly exploits short-horizon structural regularities in the sensed environment. A wall-extended local referencing mechanism is introduced to construct a virtual target ahead of the vehicle, enabling anticipatory motion while preserving a fully reactive and lightweight formulation. The approach requires no mapping, memory, or global optimization, making it suitable for resource-constrained aerial platforms. Extensive simulations and real-world experiments using a nano-UAV in modular tunnel environments demonstrate that SAMP achieves significantly improved traversal robustness compared to representative reactive baselines, particularly in scenarios involving horizontal–vertical transitions and complex junctions.

1 Introduction

Unmanned aerial vehicles (UAVs) have become widely-used for inspection and exploration in environments that are hazardous, inaccessible, or structurally complex. In particular, confined infrastructures such as tunnels, ducts, chimneys, and underground facilities present strong application demands for autonomous inspection [1, 2, 3, 4]. While ground-based platforms such as snake robots and soft robots can effectively traverse highly constrained spaces [5, 6, 7], their deployment becomes increasingly limited in long, elevated, and vertically structured tunnel networks. In contrast, micro- and nano-UAVs offer superior mobility and the ability to access multi-level structures, making them promising candidates for confined-space inspection.

Despite this potential, autonomous navigation of nano-UAVs in narrow tunnels remains fundamentally challenging. The severe size constraints of these platforms limit both sensing and onboard computation, often restricting the available perception to sparse range measurements and lightweight optical flow. At the same time, confined environments impose tight geometric constraints and complex aerodynamic interactions, including ground, ceiling, and wall effects, which significantly influence flight stability [8, 9]. These factors render conventional navigation pipelines impractical for nano-UAV systems, typically based on mapping, localization, and global planning.

Existing motion planning approaches in robotics, such as graph-based search, model predictive control, and map-based local planning, generally rely on dense sensing and computational resources [10, 11, 12, 13, 14, 15, 16]. While such methods have demonstrated success in larger UAV platforms, they are not directly applicable to resource-constrained nano-UAVs. Recent works have explored lightweight navigation strategies using minimal sensing, including optical-flow-based control and reactive obstacle avoidance [17, 18]. Reactive methods such as Bug algorithms and wall-following strategies offer computational simplicity, but they are inherently myopic, relying on immediate obstacle interactions without exploiting higher-level environmental structure. As a result, their performance degrades in tunnel environments involving junctions, vertical transitions, or extended corridors.

This paper argues that confined environments such as tunnels exhibit *implicit structural regularities* that can be exploited even under minimal sensing. Unlike general unstructured environments, tunnel-like spaces possess locally consistent geometric patterns (e.g., corridor continuity, wall alignment, and directional openings) that can provide navigational cues beyond instantaneous obstacle avoidance. However, existing lightweight planners largely fail to incorporate such structural information in a computationally efficient manner.

To address this gap, we propose a *Structure-Aware Motion Planning (SAMP)* strategy for nano-UAV navigation in confined environments using only sparse range sensing. The key idea is to introduce a short-horizon geometric interpretation of the environment by constructing a *wall-extended local*

*Email address(es): abner_asignacion@chiba-u.jp

reference ahead of the vehicle. This virtual reference point acts as a directional guide, enabling anticipatory motion while maintaining a fully reactive and lightweight formulation. Unlike conventional approaches that rely on wall contact, centerline balancing, or cautious probing, the proposed method explicitly leverages local structural cues to improve navigation consistency and robustness.

The proposed planner operates without mapping, memory, or global optimization, making it suitable for platforms with strict sensing and computational constraints. To evaluate its effectiveness, we conduct comparative simulation studies against other known motion planners. Validation is performed through both simulations and real-world experiments using a nano-UAV platform equipped with optical flow and multiranger sensors in modular tunnel environments with horizontal and vertical segments.

The main contributions of this work are summarized as follows:

- A novel **structure-aware reactive motion planning** framework for nano-UAVs operating under minimal sensing constraints.
- A **wall-extended local referencing** mechanism that enables anticipatory navigation without requiring mapping or global state estimation.
- A systematic **comparative evaluation** in confined tunnel environments, including real-world validation.

The remainder of this paper is organized as follows. Section II presents the proposed structure-aware motion planning method. Section III describes the simulation setup and evaluation framework. Section IV presents experimental results and discussion.

2 METHODOLOGY

We propose a *Structure-Aware Motion Planning (SAMP)* framework for nano-UAV navigation in confined tunnel environments using minimal range sensing. The approach is designed to operate under strict sensing and computational constraints, without relying on mapping, memory, or global optimization.

The key idea is to exploit short-horizon structural regularities in tunnel-like environments by constructing a local geometric reference ahead of the vehicle. This enables anticipatory motion while preserving a fully reactive formulation. The planner operates in two main components: (i) a structure-aware local planner based on wall-extended referencing, and (ii) a mode-switching framework for handling mixed horizontal and vertical tunnel geometries.

2.1 Structure-Aware Local Planning

Unlike conventional reactive methods such as wall following or centerline balancing, the proposed SAMP explicitly interprets the local corridor geometry to guide motion. Instead of reacting only to instantaneous obstacle measurements, the planner constructs a short-horizon *virtual reference point* (carrot point) ahead of the vehicle.

2.1.1 Frontal Structure Evaluation

To determine whether the local environment exhibits sufficient structural regularity, the controller evaluates the temporal consistency of forward range measurements. Let $\{r_F^{(i)}\}_{i=1}^{N}$ denote the recent forward measurements. Their mean and standard deviation are computed as

$$\bar{r}_F = \frac{1}{N}\sum_{i=1}^{N} r_F^{(i)}, \quad \sigma_{r_F} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(r_F^{(i)} - \bar{r}_F\right)^2}. \tag{1}$$

A structure-aware mode is activated when the forward obstacle is both within a relevant range and temporally stable:

$$d_{\text{safe}} + \delta < \bar{r}_F < r_{\max}, \qquad \sigma_{r_F} < \sigma_{\text{th}}. \tag{2}$$

This condition ensures that the detected corridor geometry is sufficiently consistent to support short-horizon extrapolation.

2.1.2 Wall-Extended Local Reference

Once activated, the planner constructs a local reference point ahead of the UAV. The forward extent is defined as

$$\Gamma = \text{clip}(r_F - d_{\text{stop}},\, 0,\, \Gamma_{\max}), \tag{3}$$

while the lateral offset is determined by the imbalance between left and right wall distances:

$$y_c = k_y(r_L - r_R). \tag{4}$$

The resulting carrot point is given by

$$(x_c, y_c) = (\Gamma, y_c). \tag{5}$$

This construction can be interpreted as a short-horizon extension of the corridor geometry, enabling the UAV to steer toward an anticipated free-space direction rather than reacting purely to local measurements.

The desired heading toward the carrot point is computed as

$$\psi = \tan^{-1}\left(\frac{y_c}{\max(x_c, \varepsilon)}\right), \tag{6}$$

and the planar velocity commands are defined as

$$v_x = v\cos\psi, \quad v_y = v\sin\psi, \tag{7}$$

with speed

$$v = \text{clip}(k_c x_c,\, 0,\, v_{\text{nom}}). \tag{8}$$

The yaw rate aligns the vehicle with the reference direction:

$$\omega_z = \text{clip}(k_\psi \psi,\, -\omega_{\max},\, \omega_{\max}). \tag{9}$$

This formulation enables smooth, anticipatory motion while maintaining computational simplicity.

http://www.imavs.org/

http://www.imavs.org/

2.1.3 Fallback and Safety Behavior

When the forward structure is not sufficiently stable, the planner reverts to a simpler reactive strategy based on forward clearance and lateral centering. Additionally, velocity commands are modulated based on proximity to obstacles:

$$v_x \leftarrow \min(v_x, v_{\text{slow}}), \quad \text{if } r_F < d_{\text{slow}}. \tag{10}$$

All commands are further constrained by instantaneous safety envelopes derived from current range measurements, ensuring collision avoidance under all conditions.

2.1.4 Mode Switching for Vertical Tunnels

To enable navigation in tunnels with vertical shafts, the proposed method extends the same structure-aware logic through a mode-switching and virtual frame formulation. The planner operates in three modes: horizontal (H), ascending (ASC), and descending (DESC). Transitions are triggered based on the availability of free space. For example, ascent is selected when forward motion is constrained while sufficient upward clearance exists:

$$r_F < d_{turn}, \quad r_U > d_{open}. \tag{11}$$

Similarly, descent is selected when downward clearance dominates. Transitions back to horizontal motion occur when vertical motion becomes constrained and horizontal openings reappear.

A key component of the proposed framework is the use of a *virtual navigation frame*. Regardless of the active mode, the progress direction is always treated as a virtual forward direction:

$$(r_F, r_B, r_L, r_R) = \begin{cases} (r_F^0, r_B^0, r_L^0, r_R^0), & \text{H}, \\ (r_U^0, r_D^0, r_L^0, r_R^0), & \text{ASC}, \\ (r_D^0, r_U^0, r_L^0, r_R^0), & \text{DESC}. \end{cases} \tag{12}$$

This allows the same structure-aware planning logic to be reused across horizontal and vertical motion without modification.

The computed velocity commands are mapped back to the physical frame by swapping the progress axis:

$$(v_x, v_z) = \begin{cases} (v_{\text{nom}},\ v_z^{\text{alt}}), & \text{H}, \\ (v_z^{\text{alt}},\ +v_{\text{nom}}), & \text{ASC}, \\ (v_z^{\text{alt}},\ -v_{\text{nom}}), & \text{DESC}. \end{cases} \tag{13}$$

Safety constraints are enforced in the physical frame to guarantee collision avoidance.

The proposed SAMP framework combines reactive control with short-horizon structural reasoning, enabling robust navigation in confined environments using minimal sensing. By integrating wall-extended referencing with virtual frame remapping, the method achieves consistent performance across complex tunnel geometries while maintaining low computational cost.

Table 1: Control parameters used in both simulation and experiment.

Parameter	Symbol	Value
Forward gain	k_x	1.2
Lateral gain	k_y	1.8
Vertical gain	k_z	1.5
Yaw gain	k_ω	0.9
Nominal velocity	v_{nom}	0.16 m/s
Maximum velocity	$v_{\max}$	0.25 m/s
Maximum yaw rate	$\omega_{\max}$	1.2 rad/s
Creep velocity	v_{creep}	0.06 m/s
Stop threshold	d_{stop}	0.18 m
Turn threshold	d_{turn}	0.24 m
Slowdown threshold	d_{slow}	0.36 m
Open-space threshold	d_{open}	0.50 m
Maximum look-ahead distance	$\Gamma_{\max}$	0.35 m
Maximum vertical dwell time	$t_{\text{nav}}^{\max}$	1.0 s
Future horizon	N	8

3 SIMULATION RESULTS

The simulation results were obtained using a MATLAB-based implementation of the proposed and baseline motion planning algorithms. The design parameters are summarized in Table 1.

The simulation environment consists of a narrow tunnel-like structure with multiple turns, representing confined inspection scenarios such as building ducts and underground passages. The tunnel has a square cross-section of $0.5\ \text{m} \times 0.5\ \text{m}$ and a total length of approximately $11\ \text{m}$. The UAV starts from an initial location outside the tunnel and is required to navigate through the corridor toward the goal while avoiding collisions with the surrounding walls.

3.1 Velocity Profile Analysis in Horizontal Navigation

Figure 1 compares the trajectories generated by the evaluated motion planners, including Rapidly-exploring Random Trees (RRT), local grid-based A* reached the goal and a lightweight bug MP from [17]. The lightweight reactive planners maintained a smooth and centered trajectory through the corridor and successfully negotiated the turns using only local sensing information. This behavior is desirable in narrow tunnel environments, where maintaining clearance from the walls is critical for safe flight.

Table 2 summarizes the quantitative performance of the planners. The lightweight reactive planner achieved the shortest completion time and the lowest computational cost while maintaining a competitive path length. In contrast, RRT and local grid-based A* required substantially longer computation times due to repeated local replanning. Standard RRT and A* implementations generally assume access to an occupancy grid or geometric map; however, such information is difficult to obtain on nano-UAVs equipped only with sparse range sensors. Therefore, in this study, RRT and A* were

Table 2: Performance comparison of motion planners in MATLAB simulation

Planner	Time Completed	Path Length	Clearance	Ave. Speed	Comp. Time (μs)	Strength/Weakness
SAMP	90 s	12.5 m	0.10 m	0.1384 m/s	63.6	Minimal sensing, Fast
Bug	88 s	12.5 m	0.05 m	0.1429 m/s	72.2	Fast but low clearance
RRT	179 s	13.3 m	0.08 m	0.0744 m/s	217.0	Slow and inefficient
A*	133 s	16.6 m	0.08 m	0.1250 m/s	47.3	Poor at corners

implemented in a receding-horizon manner, where candidate waypoints were sampled within a locally inferred free-space corridor derived from range measurements. Instead of explicitly constructing a map, tunnel boundaries were imposed as feasibility constraints. Although this modification allowed both planners to progress toward the goal, their computational burden remained significantly higher than that of the lightweight reactive planner.

Figure 1 also shows the forward (v_x) and lateral (v_y) velocity profiles for representative planners. These profiles provide insight into the control behavior and motion characteristics of each method under confined navigation. These observations are worthy to note:

- **SAMP (Proposed)** The proposed SAMP exhibits a smooth and structured velocity profile. The forward velocity v_x remains close to the nominal value for most of the trajectory, with only brief reductions near turns or constrained regions. The lateral velocity v_y shows gradual and continuous variations, corresponding to anticipatory steering toward the local reference (carrot point). This behavior reflects the structure-aware nature of the planner: instead of reacting abruptly to obstacles, the UAV adjusts its motion based on short-horizon geometric inference. As a result, SAMP produces smooth transitions through corners and maintains stable forward progress with minimal oscillation.
- **Bug** In contrast, the Bug-based planner exhibits highly oscillatory lateral velocity v_y, characterized by rapid switching between positive and negative values. This behavior is a direct consequence of wall-following logic, where the controller repeatedly corrects its position relative to the wall without anticipating future geometry. The forward velocity v_x also shows intermittent drops, particularly near corners, indicating hesitation and inefficient progress. These oscillations lead to increased path inefficiency and reduced stability, especially in narrow passages where frequent corrections are required.
- **RRT** The RRT-based planner produces irregular and discontinuous velocity profiles. Both v_x and v_y exhibit sudden changes and spikes, reflecting the discrete and replanning-based nature of the algorithm. As the planner generates new waypoint sequences, the control commands change abruptly, resulting in non-smooth motion. Although RRT is able to reach the goal, the velocity profile indicates poor suitability for real-time control on nano-UAVs, as the lack of continuity can introduce instability and increased control effort.
- **Local Grid A*** Similarly, the local grid-based A* planner shows high-frequency fluctuations in both v_x and v_y, particularly during replanning phases. Large bursts in lateral velocity are observed when the planner updates its path, causing abrupt directional changes. These discontinuities arise from the grid-based representation and frequent local replanning, which do not explicitly enforce motion smoothness. While the planner successfully reaches the goal, the resulting motion is less stable and more computationally demanding compared to lightweight reactive approaches.

Overall, the velocity profiles highlight a key advantage of the proposed SAMP: it achieves smooth, continuous control behavior while maintaining reactivity and low computational complexity. In contrast, Bug suffers from oscillatory corrections due to its purely local nature, while RRT and A* introduce discontinuities due to replanning. These results demonstrate that incorporating short-horizon structural awareness enables more stable and efficient motion in confined environments under minimal sensing.

3.2 Velocity Profile Analysis in Mixed Tunnel Navigation

The velocity profiles in Fig. 2 further highlight the difference in motion behavior between SAMP and the Bug-based planner, particularly in the presence of vertical transitions.

The proposed SAMP exhibits smooth and coordinated velocity profiles across all three axes. The forward velocity v_x remains close to the nominal value, with only moderate reductions near turns and transition regions. Hence, the lateral velocity v_y varies gradually, reflecting continuous steering toward the wall-extended local reference. Importantly, the vertical velocity v_z shows structured and intentional activation only during vertical transitions. The transitions between horizontal and vertical motion are smooth and temporally consistent, indicating that the planner can anticipate and commit to vertical motion based on local geometric cues. This results in stable traversal of vertical segments without oscillatory switching.

In contrast, the Bug-based planner exhibits highly discontinuous velocity profiles in all axes. The lateral velocity v_y shows rapid switching behavior, corresponding to re-

http://www.imavs.org/

http://www.imavs.org/

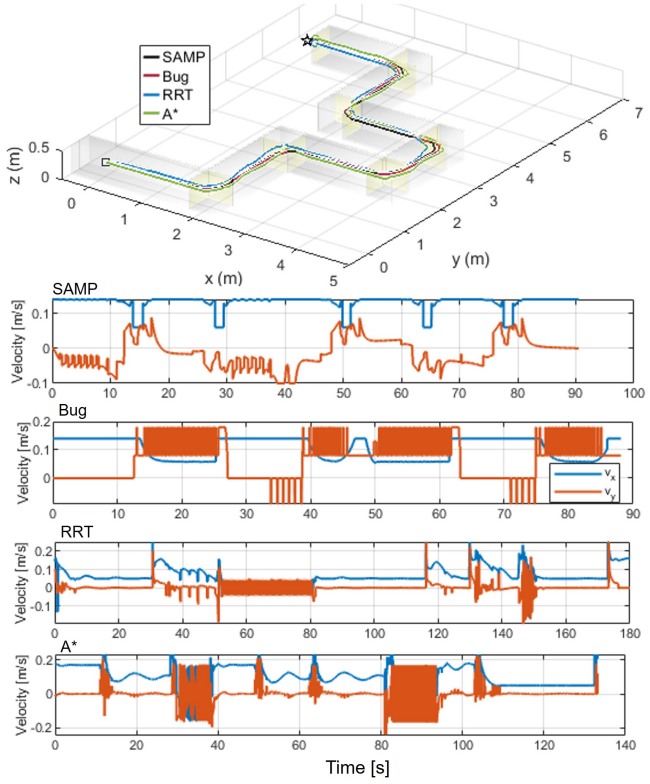


Figure 1: Comparison of motion planners in the MATLAB tunnel simulation environment. The proposed structure-aware motion planning (SAMP) and the Bug-based planner are highlighted as lightweight methods suitable for nano-UAV deployment. SAMP achieves a smoother and more centered trajectory through the corridor, while the Bug planner exhibits oscillatory behavior near turns due to its purely reactive nature.

peated wall-following corrections. This oscillatory behavior becomes more pronounced near corners and junctions. The vertical velocity v_z exhibits frequent sign changes and intermittent activation, indicating instability in selecting and maintaining vertical motion. Rather than committing to a clear ascent or descent, the planner repeatedly switches between directions, leading to inefficient and prolonged traversal. The forward velocity v_x also shows frequent interruptions, reflecting hesitation and reduced progress due to the lack of anticipatory behavior.

These results demonstrate that purely reactive strategies such as Bug are insufficient for mixed horizontal–vertical tunnel navigation, where consistent mode commitment is required. In contrast, SAMP leverages short-horizon structural awareness to generate smooth, coordinated motion across all axes. The resulting velocity profiles are significantly more stable, enabling reliable traversal of complex confined environments under minimal sensing.

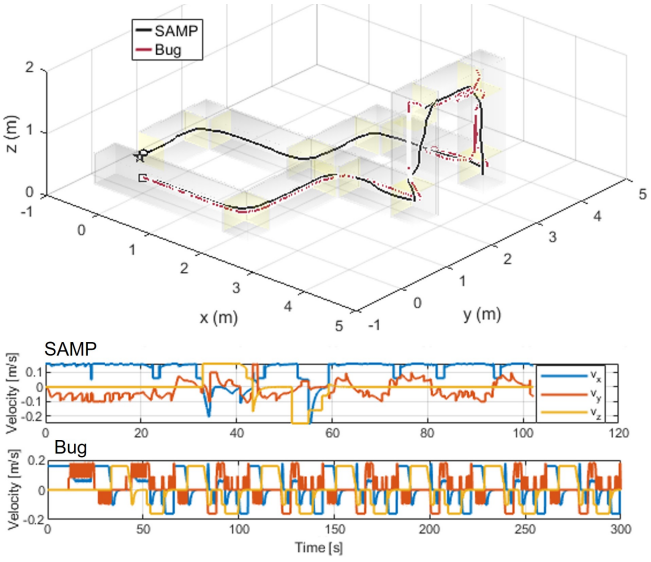


Figure 2: Comparison between the proposed structure-aware motion planning (SAMP) and a Bug-based reactive planner in a mixed horizontal–vertical tunnel environment. (Top) 3D trajectories from start to goal. SAMP achieves smooth and consistent traversal, including stable vertical transitions, while the Bug planner exhibits oscillatory behavior and inefficient motion, particularly near corners and vertical junctions. (Bottom) Corresponding velocity profiles in the forward (v_x), lateral (v_y), and vertical (v_z) directions. SAMP maintains smooth and continuous velocity variations, whereas the Bug planner shows frequent switching and high-frequency oscillations due to its purely reactive wall-following behavior.

4 Experimental Validation

Real-world flight experiments were conducted using a Crazyflie 2.1 nano-UAV to evaluate the proposed lightweight motion planners under minimal sensing conditions. The platform has a mass of approximately 28 g and a motor-to-motor distance of 0.08 m. The UAV is equipped with onboard IMU and pressure sensors, together with a Multiranger deck and Flow deck for sparse range sensing, optical-flow-based velocity estimation, and altitude stabilization. No global map, motion capture system, 3D LiDAR, or vision-based localization was used during autonomous navigation.

Figure 3 shows the experimental tunnel environments and representative flight trajectories. The navigation task in all experiments was to autonomously traverse the tunnel and reach the end of the course using only onboard sparse sensing. Two tunnel configurations were considered: a horizontal-to-ascending tunnel transition (Environment A, 2.0 m) and a mixed horizontal-vertical tunnel (Environment B, 3.5 m). Figures 3(A.2) and (B.2) compare representative trajectories generated by SAMP and the Bug-based planner. In both environments, SAMP maintained smoother and more centered trajectories, while the Bug-based planner exhibited larger de-

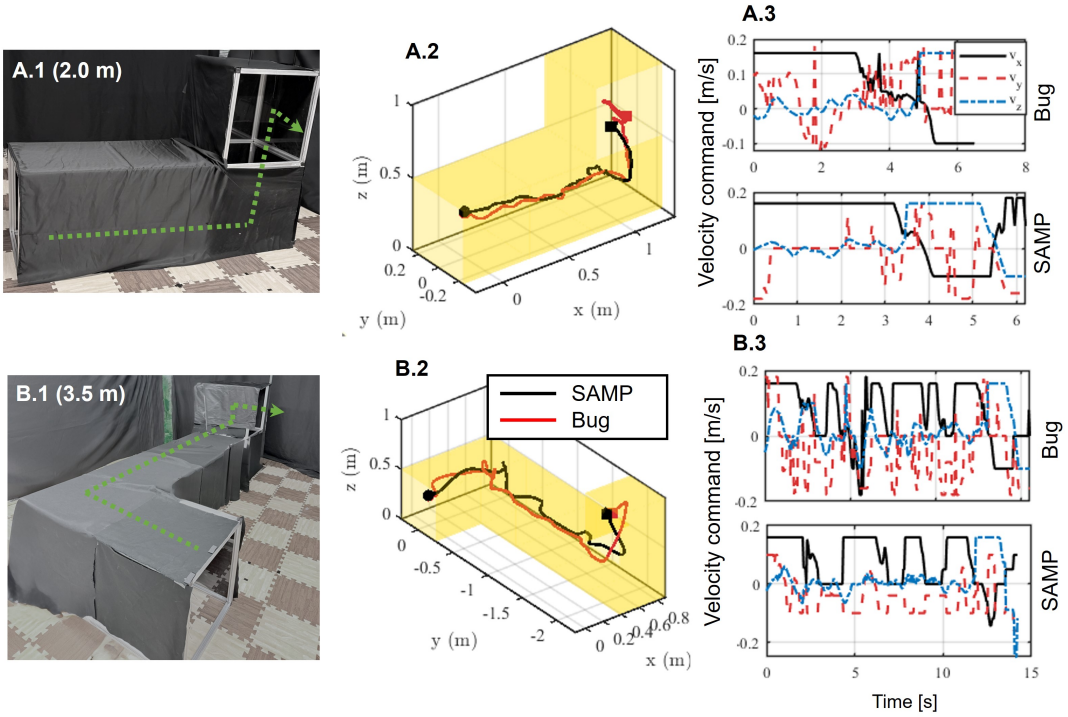


Figure 3: Experimental validation of lightweight motion planners in confined tunnel environments using a Crazyflie nano-UAV equipped with sparse onboard sensing. (A.1) Horizontal-to-ascending tunnel environment. (A.2) Representative experimental trajectories for SAMP and the Bug-based planner in Environment A. (A.3) Corresponding velocity profiles showing forward (v_x), lateral (v_y), and vertical (v_z) motion components. (B.1) Mixed vertical-horizontal tunnel environment. (B.2) Representative experimental trajectories for SAMP and the Bug-based planner in Environment B. (B.3) Corresponding velocity profiles for the evaluated planners. The proposed SAMP maintains smoother trajectories and more continuous velocity behavior, whereas the Bug-based planner exhibits stronger oscillatory corrections due to purely reactive wall-following behavior under sparse sensing.

viations and oscillatory corrections, particularly near tunnel transitions and corners.

The corresponding velocity profiles in Figs. 3(A.3) and (B.3) further highlight the behavioral differences. SAMP produced smoother forward, lateral, and vertical velocity transitions, whereas the Bug-based planner showed frequent oscillations caused by purely reactive wall-following behavior without anticipation of tunnel geometry.

The experimental success rates also demonstrate the advantage of incorporating short-horizon structural awareness. In Environment A, the Bug-based planner achieved a success rate of $5/10$, while SAMP achieved $8/10$. In the more challenging Environment B, Bug achieved only $1/10$ successful runs, whereas SAMP achieved $6/10$. Failures were mainly caused by aerodynamic disturbances inside the tunnels, imperfect range measurements, and local interpretation errors. Some runs resulted in wall collisions, while others caused the UAV to retreat toward previously traversed regions instead of progressing toward the goal.

Overall, the experimental results demonstrate that incorporating lightweight structural awareness significantly improves navigation robustness and motion smoothness for nano-UAV tunnel navigation under sparse sensing conditions.

5 Conclusion

This paper presented a Structure-Aware Motion Planner (SAMP) for nano-UAV navigation in confined tunnel environments using only sparse range sensing. The proposed approach introduces a wall-extended local referencing mechanism that enables anticipatory yet lightweight reactive navigation without requiring mapping, memory, or global optimization. Simulation and real-world experiments demonstrated that SAMP achieves smoother motion, improved stability, and higher traversal success rates compared to conventional reactive baselines. Overall, this work demonstrates the feasibility of autonomous tunnel navigation for resource-constrained nano-UAVs using minimal onboard sensing and highlights the potential of lightweight structure-aware planning for confined and GPS-denied environments.

http://www.imavs.org/

http://www.imavs.org/

ACKNOWLEDGEMENTS

This work was supported by the New Energy and Industrial Technology Development Organization (NEDO) under Project JPNP22002.

REFERENCES

[1] Jan Quenzel, Matthias Nieuwenhuisen, David Droeschel, Marius Beul, Sebastian Houben, and Sven Behnke. Autonomous mav-based indoor chimney inspection with 3d laser localization and textured surface reconstruction. *Journal of Intelligent & Robotic Systems*, 93(1):317–335, 2019.

[2] Chee How Tan, M Ng, Danial Sufiyan Bin Shaiful, Shane Kyi Hla Win, WJ Ang, Sai Kit Yeung, HB Lim, Minhn N Do, and S Foong. A smart unmanned aerial vehicle (uav) based imaging system for inspection of deep hazardous tunnels. *Water Practice & Technology*, 13(4):991–1000, 2018.

[3] B. He, X. Ji, G. Li, and B. Cheng. Key technologies and applications of uavs in underground space: A review. *IEEE Transactions on Cognitive Communications and Networking*, 10(3):1026–1049, 2024.

[4] R. Zhang, G. Hao, K. Zhang, and Z. Li. Unmanned aerial vehicle navigation in underground structure inspection: A review. *Geological Journal*, 58(6):2454–2472, 2023.

[5] Xuesu Xiao and Robin Murphy. A review on snake robot testbeds in granular and restricted maneuverability spaces. *Robotics and Autonomous Systems*, 110:160–172, 2018.

[6] Sangchul Han, Sanguk Chon, JungYeong Kim, Jeahong Seo, Dong Gwan Shin, Sangshin Park, Jin Tak Kim, Jinhyeon Kim, Maolin Jin, and Jungsan Cho. Snake robot gripper module for search and rescue in narrow spaces. *IEEE Robotics and Automation Letters*, 7(2):1667–1673, 2022.

[7] Peter Wharton, Tsam Lung You, George P Jenkinson, Richard Suphapol Diteesawat, Nguyen Hao Le, Edith-Clare Hall, Martin Garrad, Andrew T Conn, and Jonathan Rossiter. Tetraflex: A multigait soft robot for object transportation in confined environments. *IEEE Robotics and Automation Letters*, 8(8):5007–5014, 2023.

[8] Darius J Carter, Lauren Bouchard, and Daniel B Quinn. Influence of the ground, ceiling, and sidewall on microquadrotors. *AIAA Journal*, 59(4):1398–1405, 2021.

[9] Pedro Sanchez-Cuevas, Guillermo Heredia, and Anibal Ollero. Characterization of the aerodynamic ground effect and its influence in multirotor control. *International Journal of Aerospace Engineering*, 2017(1):1823056, 2017.

[10] Matěj Petrlík, Tomáš Báča, Daniel Heřt, Matouš Vrba, Tomáš Krajník, and Martin Saska. A robust uav system for operations in a constrained environment. *IEEE Robotics and Automation Letters*, 5(2):2169–2176, 2020.

[11] Mihail Pivtoraiko, Daniel Mellinger, and Vijay Kumar. Incremental micro-uav motion replanning for exploring unknown environments. In *2013 IEEE international conference on robotics and automation*, pages 2452–2458. IEEE, 2013.

[12] Danilo Saccani, Leonardo Cecchin, and Lorenzo Fagiano. Multitrajectory model predictive control for safe uav navigation in an unknown environment. *IEEE Transactions on Control Systems Technology*, 31(5):1982–1997, 2022.

[13] Xinhua Wang, Mingyan Cheng, Shuai Zhang, and Huajun Gong. Multi-uav cooperative obstacle avoidance of 3d vector field histogram plus and dynamic window approach. *Drones*, 7(8):504, 2023.

[14] Shengjie Fu, Chang Lin, Zhaoyuan Yao, Tianliang Lin, and Qihuai Chen. Obstacle-avoidance trajectory planning method for excavators in confined spaces based on improved rrt* algorithm. *Scientific Reports*, 2026.

[15] Mehak Raibail and Abdul Hadi Abd Rahman. Path planning of multi-robots in confined spaces. In *International Conference on Robot Intelligence Technology and Applications*, pages 200–208. Springer, 2023.

[16] Hui Ye, Junjie Cao, Xiaofei Yang, and Shuyi Shao. Safety-critical model predictive control for quadcopter uav subject to wind disturbances and measurement errors in confined environments. *Aerospace Science and Technology*, page 111790, 2026.

[17] Kimberly N. McGuire, Christophe De Wagter, Karl Tuyls, Hilbert J. Kappen, and Guido C. H. E. de Croon. Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. *Science Robotics*, 4(35):eaaw9710, 2019.

[18] Kimberly N. McGuire, Guido C. H. E. De Croon, Christophe De Wagter, Karl Tuyls, and Hilbert J. 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.

Downscaling Autonomous Navigation of Tiny Drones under Dense Forest Canopies

Andreas Zwanenburg, Salua Hamaza
BioMorphic Intelligence Lab, Dept. of Control & Operations, Faculty of Aerospace Engineering,
TU Delft, The Netherlands

http://www.imavs.org/

Abstract

This study investigates the effects of downscaling hardware and software on the performance and processor load of small drones, with a focus on rainforest exploration. Through systematic benchmarking, we evaluate state-of-the-art path-planning on resource-limited platforms, examining trade-offs between computational load and key navigation metrics including gap size, obstacle detection, and reaction time. We offer practical guidelines for autonomous navigation on low-power processors in dense, narrow-gap environments. Finally, we demonstrate the feasibility of this approach through a 100-gram prototype achieving full onboard autonomous navigation through unknown, unstructured forest environments.



Figure 1: The 100-gram autonomous drone (DJI Tello with Raspberry Pi Zero 2W, and Arducam ToF camera) navigating in a dense forest.

1 Introduction

Tropical rainforests, essential to the planet's ecological health, are increasingly threatened by deforestation and climate change [1]. Preserving these ecosystems demands a deeper understanding of their biodiversity, a task complicated by the sheer size and inaccessibility of rainforest environments. In recent years, drones have emerged as a promising solution, enabling researchers to monitor forest conditions, gather critical biodiversity data, and evaluate the effects of climate change and human activity, as shown by [2–6]. A key advantage of drones lies in their ability to navigate remote, hard-to-reach areas with relative ease [7, 8].

Advances in aerial robotics have greatly expanded the autonomy of unmanned aerial vehicles (UAVs). The EGO-Planner, introduced by Zhou et al. [9], marked a milestone in making high-performance local planning feasible on small quadrotors, while recent work demonstrated vision-based route following indoors on a 56-gram drone [10], hinting at the potential for outdoor use. However, that system is limited to retracing previously flown routes in obstacle-free conditions, whereas our system performs fully autonomous navigation through unknown, unstructured environments. Such route-following approaches, while highly memory-efficient, require prior traversal of the environment and cannot handle unknown obstacles — capabilities that are essential for rainforest exploration, where terrain is unstructured and no prior map is available. Yet many state-of-the-art obstacle-avoiding algorithms still depend on high-performance processors and sensors, adding significant weight and reducing endurance. For instance, the EGO-Planner implementation employs a 150-gram microprocessor and a 40-gram stereo depth camera on a 900-gram platform, leading to a take-off weight of over 1 kg and flight time of 8 minutes. Lighter onboard processors and sensors directly extend flight endurance by reducing payload mass; this also enables the use of smaller, cheaper, and safer drone platforms that achieve comparable endurance to heavier systems, making them more practical for real-world applications.

Within this work, our objective is to advance the understanding of ultra-lightweight autonomous drones and their performance in complex natural environments. We present a fully autonomous aerial system weighing only 100 g, with all computations performed onboard. Its lightweight design enables agile navigation in cluttered environments such as dense tropical forests, addressing critical challenges of rainforest deployment. We systematically analysed the computational demands of path planning, establishing performance metrics to evaluate navigation effectiveness. Benchmarking tests measured computer processor load, memory usage, and loop durations across different configurations, while simulation experiments incorporating real-forest data allowed us to explore the relationship between navigation performance, CPU load, and sensor data quality. This process identified computational bottlenecks within path planning sub-processes, of-

http://www.imavs.org/

fering insights into power optimization. This paper begins by describing our baseline 500-gram rainforest drone, developed for the XPRIZE Rainforest competition, which served as the starting point for our downscaling experiments.

The main contributions of this work are:

- A systematic benchmarking framework for evaluating the trade-offs between computational load and navigation performance in resource-constrained path planning, using four defined metrics: minimum solvable gap size, minimum detectable obstacle size, solvable obstruction width, and reactiveness time.
- A Pareto-front analysis revealing feasible configurations for autonomous navigation under processor constraints, providing practical guidelines for system designers.
- A proof-of-concept implementation of autonomous 3D obstacle mapping and path planning on a 100-gram drone platform, demonstrating that real-time autonomous navigation through unknown, unstructured outdoor environments is feasible at this scale.

2 Autonomous Flight with a 500-Gram Drone

A 500-gram autonomous drone was developed in the context of the XPRIZE Rainforest Competition [11], which tasks teams with surveying large forest areas without human entry. To address this challenge, we derived a set of technical requirements: fully autonomous flight without external signals (GPS, joystick, or video stream), robust navigation through dense vegetation with obstacles at all heights, and sufficient flight time to cover meaningful ground area. The platform is a Parrot Bebop 2 drone equipped with an Intel RealSense D435 depth camera for obstacle sensing and an ODROID-C4 computer for onboard autonomy. Low-level flight control is handled by the Bebop 2's internal controller, interfaced via the Robot Operating System (ROS) using the bebop_autonomy SDK [12].

2.1 Energy Usage Benchmarking

When downscaling a drone to a smaller form factor, energy consumption becomes one of the biggest bottlenecks. With a smaller battery, smaller motors and smaller propellers, the payload that the drone can handle becomes more critical. Sub-100-gram drones typically have a flight time of approximately 5 minutes. But with even the smallest addition of mass, the available flight time drops significantly. To integrate an autonomous path planner on tiny drones, extra sensors such as a depth camera and a processor unit are required. To optimise path planning in forest conditions on resource-limited processors, benchmark tests are performed to measure CPU and memory load. To interpret these benchmarking results meaningfully, we define a set of performance metrics that capture both navigation safety and computational demand.

Performance Metrics

Autonomous planning involves perception and planning steps. This section introduces four metrics created for safe and robust navigation in dense environments: reactiveness time, solvable obstruction width, minimum detectable object size, and minimum solvable gap size. Experiments on a 500-gram drone indicate that processing load is more affected by perception configurations than planning configurations. Depth image rate and resolution significantly influence processing load. Planner configurations, like cost function and maximum iterations, impact performance but do not significantly alter processor load, therefore these are not considered in the trade-off comparison.

Depth Image Projection

Depth image projection translates 2D depth images into 3D detection points. This process involves using intrinsic and extrinsic camera matrices to determine the 3D position of every pixel with respect to the camera's coordinate frame, illustrated in Figure 2 [13].

Our benchmarking explores how lowering image resolution and frame rate affects perception performance, CPU load and memory load. Once points are projected into 3D, we must update the map to reflect both obstacles and free space—this is handled by raycasting.

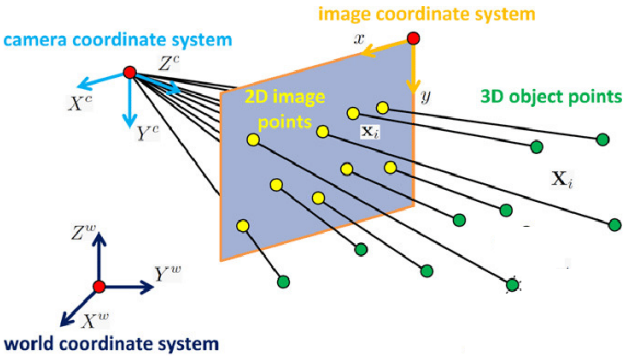


Figure 2: Illustration of depth image projection from a 2D camera frame into a 3D obstacle map, based on camera intrinsics and extrinsics [13].

Raycasting Process

Raycasting updates the 3D map by considering obstacles detected by the drone. The probability of currently mapped obstacles that lie in between the drone and newly detected obstacles will be lowered, applying the probabilistic occupancy mapping concept [14]. It assumes no obstacles lie between the drone and the detected points, crucial for updating the obstacle map. Raycasting addresses sensor noise from the camera, essential to creating a robust obstacle map in challenging environments. With the map updating in real time, we can now quantify what sizes of gaps the drone can navigate.

http://www.imavs.org/

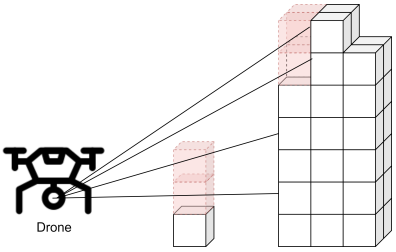


Figure 3: Illustration of the raycasting process used in occupancy mapping. Voxels marked as obstacles but lying between the drone and new detections (red blocks) are removed to reduce false positives and sensor noise.

Minimum Solvable Gap Size

To find path-planning solutions in a dense forest, identifying small gaps is crucial. Therefore, we created the minimum gap size metric, which can be used to determine the sizes of gaps that can be detected. The minimum gap size depends mainly on voxel grid resolution, with smaller voxels improving accuracy but significantly increasing processor and memory load — for example, reducing voxel size from 0.20 m to 0.05 m can increase memory requirements by tens of times, depending on environment structure and octree pruning, as illustrated in Figure 4, observed in our benchmarking, and consistent with voxel-based frameworks like OctoMap [15].

This metric considers factors like pixel size, camera resolution, inflation distance, and grid resolution. Hypotheses suggest CPU load increases with decreasing gap sizes, impacting the performance of the algorithm.

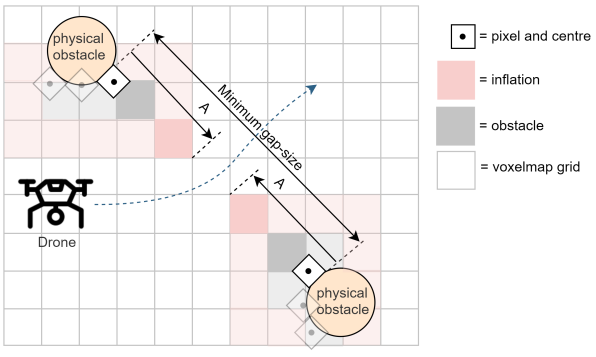


Figure 4: Illustration of the minimum solvable gap size for diagonal gaps in voxel-based mapping. The resolution of the voxel grid determines the smallest gap that can be reliably detected.

Equation 1 defines the guaranteed minimum gap size based on configurations and camera specifications. In the equation, PS is the pixel size, FOV the camera's field of view, R_{cam} the camera range, C_{res} the camera resolution, D_{infl} the inflation distance, D_{min} the minimum inflation distance, GR the voxel grid resolution, and GS_{min} the minimum solvable gap size. A refers to the dimensioning illustrated in Figure 4. The square brackets, also known as the ceiling brackets, indicate that the decimal number inside should be rounded up to the nearest whole number.

$$
\begin{aligned}
PS &= \frac{\tan\left(\frac{FOV}{2}\right) \cdot R_{cam} \cdot 2}{C_{res}} \\
D_{infl} &= \lceil \frac{D_{min}}{GR} \rceil \cdot GR \\
A &= \frac{PS}{2} + \sqrt{2 \cdot (GR)^2} + \sqrt{2 \cdot (D_{infl})^2} \\
GS_{min} &= A \cdot 2 + \sqrt{2 \cdot (GR)^2} \\
GS_{min} &= PS + 3 \cdot \sqrt{2 \cdot (GR)^2} + 2 \cdot \sqrt{2 \cdot (D_{infl})^2}
\end{aligned}
\tag{1}
$$

Depth Image Resolution

Lowering the depth image resolution, either by using a lower-quality camera, or by skipping pixels defined in the configuration, negatively impacts the detection of small obstacles. Thereby creating a risk of not detecting tiny branches in the forest and crashing. Figure 5 illustrates the potential blind zones for a certain configuration and distance to an object. But, at the same time, lowering the number of computations for the projection, thereby lowering the processor load. In benchmarking, the depth camera resolution is varied to assess its impact on drone perception performance.

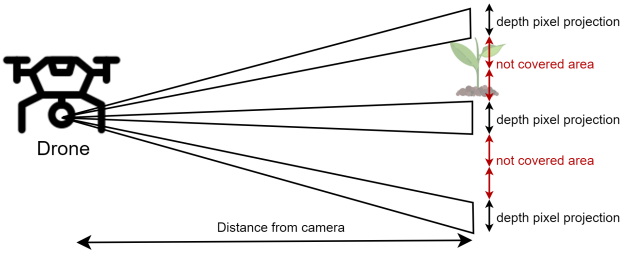


Figure 5: Illustration of pixel skipping in the projection step, where 2 out of every 3 pixels are ignored. This reduces processor load but increases blind zones and the risk of missing thin obstacles.

To estimate the minimal size of a potentially missed object at maximum camera distance, Equation 2 provides an approximation. This equation considers parameters such as FOV field of view, CR_{max} camera range, OR original resolution, and SP skipped pixels. The result, denoted as SM_{max}, represents the maximum object size that may be overlooked. The pixel projection scales linearly with the camera range, allowing detection of smaller objects at closer distances.

$$
\begin{aligned}
PP_{max} &= \frac{\tan\left(\frac{\text{FOV}}{2}\right) \cdot CR_{max} \cdot 2}{OR} \\
SM_{max} &= SP \cdot PP_{max}
\end{aligned}
\tag{2}
$$

http://www.imavs.org/

Local Planner Range

The EGO-planner's local range is the area around the drone in which obstacles are considered to plan a path around. By only using a local region, the path planner can calculate trajectories more efficiently. When the obstacle is wider than the local planner range, and the planner assumes all space outside the local planner range is obstacle-free, an issue arises. Figure 6 illustrates this issue where the planner gets stuck in a loop where it never succeeds in navigating to the goal. A well-known failure mode of local, potential-field–style planners, a limitation first noted in classic potential-field navigation studies. A smaller range reduces CPU load but may hinder efficient route planning. Balancing local planner range is crucial to avoid navigation loops and ensure effective obstacle avoidance.

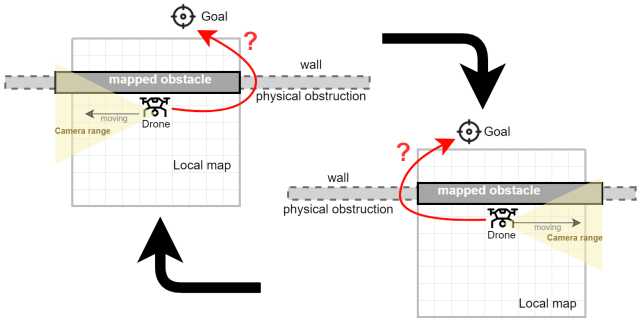


Figure 6: Illustration of the local planner range limitation. When obstructions are wider than the local planning horizon, the planner incorrectly assumes free space beyond the map and may enter an infinite replanning loop.

Equation 3 can be used to determine what obstruction width can be solved by the algorithm. In the equation, D_{infl} is the inflation distance, D_{min} the minimum inflation distance, GR the voxel grid resolution, R_{loc} the local range, and SOW the solvable obstruction width.

$$\begin{aligned} D_{infl} &= \lceil \frac{D_{min}}{GR} \rceil \cdot GR \\ SOW &= R_{loc} - 2 \cdot D_{infl} \end{aligned} \tag{3}$$

Reactiveness

Reducing the depth image rate, by using a lower-quality camera or skipping images by configuration, the processor load can be lowered. But, for newly occurring obstacles, it takes more time to map and detect a possible collision. By varying the depth image rates during benchmarking, the processor load and reactiveness are compared.

Equation 4 shows what sub-processes and configurations affect the theoretical reactiveness duration. This is the duration between the occurrence of a new obstacle, up to the moment it is mapped and a collision check is performed. In this equation, PI is the number of iterations until the threshold is reached, P_t and P_i are respectively the probability threshold and probability increase, R_{cam} the camera rate, and T_{proj}, T_{ray}, T_{upd}, T_{col} the loop times for projection, raycasting, updating, and collision check.

$$\begin{aligned} PI &= \lceil \frac{P_t}{P_i} \rceil \\ T_{react} &= PI \cdot \frac{1}{R_{cam}} + T_{proj} + T_{ray} + T_{upd} + T_{col} \end{aligned} \tag{4}$$

2.2 *Configuration Variations*

Four parameters are varied to compare planner performance by the metrics mentioned above. These are voxel grid resolution, local map range, camera resolution, and camera rate. Each parameter variation explores its impact on the processor and memory load, offering insights into optimal configurations for autonomous flight in challenging environments. Table 1 shows per metric what parameter combinations are benchmarked.

Table 1: Simulation Experimental Setup

	Voxelgrid res. [m]	**Local map range [m]**	**Camera res. [-]**	**Camera rate [Hz]**
Gap size	[.05 .10 .15 .20]	10	[1 1/2 1/3 1/4]	15
Obstacle size	0.10	10	[1 1/2 1/3 1/4]	15
Obs. width	[.05 .10 .15 .20]	[1 2 ... 35]	1	15
Reaction time	0.10	10	1	[15 7.5 5 3.8 3]

3 Dataset Collection

Creating a dataset for simulations involves several critical steps, from data collection to processing. Initially, ROS nodes are configured to output loop durations for specific sub-processes, providing valuable insights into performance. Ground-truth data is also gathered during dataset creation to enable comparison of different methods. This section details the process of generating an outdoor dataset.

1) Data Collection Setup: To monitor the performance of the EGO-planner algorithm, ROS publishers are integrated to output data related to the duration of key sub-processes: projection, raycasting, map updates, and collision-checking for each processed depth image. This data is logged for later analysis.

2) Processor and Memory Load Monitoring: To evaluate the processor load and memory usage, the `cpu_monitor` ROS package [16] is used, publishing system metrics at a 10

http://www.imavs.org/

Hz interval. This helps assess the computational resources consumed by the algorithm during operation.

3) Evaluation of State Estimation Methods: The study evaluates a drone path-planning algorithm based on dead reckoning by conducting performance tests in both indoor and outdoor environments, each requiring distinct data collection methods. For indoor testing, high-precision motion data is obtained using the Optitrack Motion Capture System, which integrates seamlessly with the ROS network via MAVros. Outdoor tests take place in dense forest settings, where GPS data provides the ground-truth position. A UBLOX F9P GPS module equipped with a helix antenna is mounted on the drone to enable comparison between dead reckoning and GPS-based navigation.

In both scenarios, inertial measurement unit (IMU) and image data are collected to enhance position estimation through visual-inertial odometry. This is implemented using VINS-Fusion [17], built on VINS-Mono [18] and leveraging techniques also common to ORB-SLAM2 [19], which tracks landmarks in infrared images similarly to the EGO-planner [9]. Together, these approaches allow the study to comprehensively assess the effectiveness of the dead reckoning-based path-planning method across varied environmental conditions by leveraging motion capture, GPS, and visual-inertial odometry data.

4) Outdoor Dataset Collection: The outdoor dataset is recorded in a dense forest environment resembling rainforest conditions. During autonomous flight, the drone navigates a walking trail with obstacles, capturing sensor data throughout the flight. This dataset ensures repeatability, using real-forest sensor data, and minimises the need for further outdoor testing to validate algorithm configurations [20]. Our 3D mapping pipeline is conceptually aligned with recent under-canopy photogrammetric work using stereo vision to generate dense point clouds for parameter extraction [21].

5) Simulation Benchmarking: To assess the drone path-planning algorithm, simulations are run using different configurations while replaying the sensor data from the dataset. This section outlines the process of preparing datasets, running simulations, and extracting results for analysis. The dataset is curated by filtering and trimming the logged data from real flights, focusing on the drone's IMU data and the depth and grayscale images captured by the camera.

For efficient simulation and data collection, a bash script automates the process, running the planner algorithm on the dataset with various configurations. The results are then processed into Python data for further analysis. This automated approach allows for batch processing of over 100 simulations, making it a time efficient method for bulk analysis.

4 Analysis

The analysis script processes all data using specific function definitions for calculating performance metrics as described in Section 2.1. This includes pixel size, minimum gap size, maximum obstruction width, minimum obstacle size, and reactiveness.

When downsizing computational load for tiny drones, it is crucial to closely evaluate the CPU load profile during simulations. The perception phase, focused on updating the local map using depth images, differs from the planning phase, marked by peak loads during path sampling, optimization, and refinement. Figure 7 exemplifies the CPU load distribution during a simulation run with varying voxel grid resolutions. In this example, left of the vertical dotted line is only perception. While on the right side of the dotted line, the perception and path planning are both active. For the benchmarking, the path planning is triggered to re-plan every 2 seconds, causing the peaks in the CPU load.

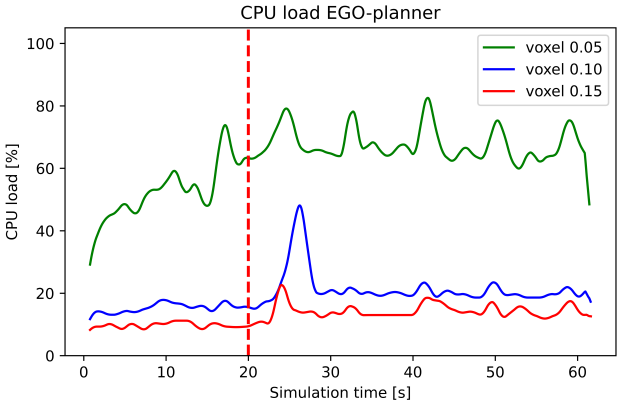


Figure 7: CPU load distribution over the benchmark simulation with voxel-grid resolutions of 0.05, 0.10, and 0.15 m.

5 Results

This section presents the benchmarking results for each of the four performance metrics defined in Section 2.

5.1 Gap Size Performance

CPU load was measured for varying voxel sizes and image resolutions. The relationship is non-linear, with smaller voxels increasing load. Figure 8 shows how grid resolution affects CPU load and the ability to navigate narrower gaps. Memory usage follows the same trend, increasing significantly with resolution; reducing voxel size from 0.20 m to 0.05 m can increase memory requirements by tens of times, depending on pruning and environment structure. This indicates that higher resolution improves gap navigation but at steep computational cost—critical for small onboard processors.

5.2 Obstacle Size Performance

For detecting smaller obstacles, the relation is tested with decreasing CPU load during the benchmarking. This test involves varying the depth image resolution and voxel grid resolution. Results, shown in Figure 9, indicate that at the

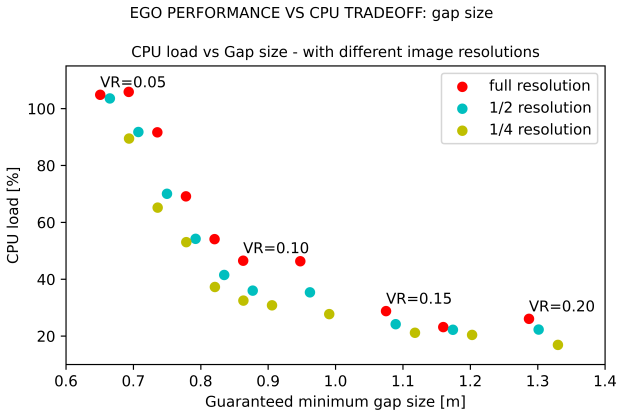


Figure 8: CPU load (%) vs. minimum solvable gap size (m) at full, half, and quarter image resolution.

finest voxel resolution tested (5 cm), relaxing the minimum detectable obstacle size leads to a significant, non-linear reduction in CPU load. At coarser voxel resolutions (7–15 cm), the same relaxation yields only a modest ∼5% reduction, which is not considered significant. This suggests that detecting only larger obstacles yields modest computational savings unless voxel resolution is already high.

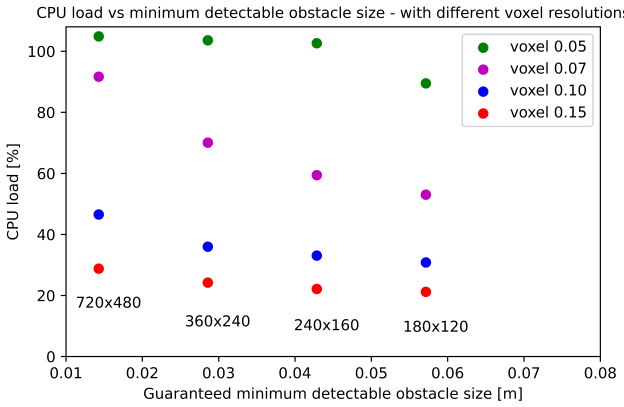


Figure 9: CPU load (%) vs. minimum detectable obstacle size (m) for voxel resolutions of 0.05 m, 0.07 m, 0.10 m, and 0.15 m.

5.3 *Obstruction Width Performance*

As mentioned earlier, path planning around wide obstructions requires a larger range for the local planner. In the benchmark, CPU load was tested for increasing local planner ranges and varied across different voxel grid resolutions. The measured loads exhibit a roughly linear increase as the obstruction width that can be handled grows, see Figure 10.

For high voxel resolutions, such as 10 cm, CPU load rises significantly—from about 35% for a 2 m wide obstruction to 85% for a 35 m wide obstruction. In contrast, for lower voxel resolutions, such as 20 cm, the load increases by only 5–10% across the same range, which is considered insignificant.

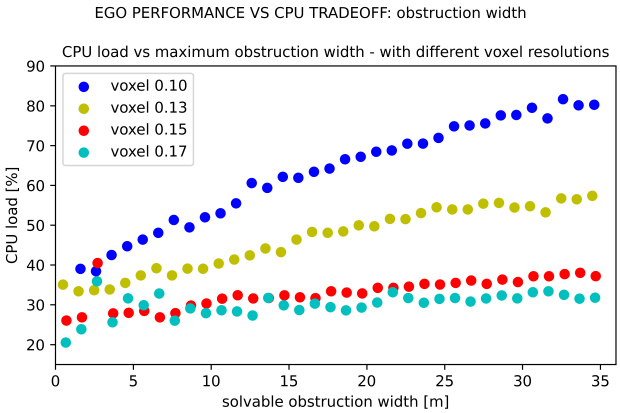


Figure 10: CPU load for various minimal obstacle widths with voxel resolutions 0.10 m, 0.13 m, 0.15 m, and 0.17 m.

5.4 *Reactive Performance*

To test the potential load increase for a system that has a quicker reaction to newly occurring obstacles, the rate of the depth images is varied in the benchmark tests from 3 to 15 depth images per second. The voxel grid resolution is also varied to increase the general system load to check if this differs in the results. The results in Figure 11 show a slightly curved decrease in load when the image rate is lowered. The plot shows that for a system with high load, the impact of decreasing the rate has significantly more impact (25% reduction) compared to the test where the general system load was low (5% reduction).

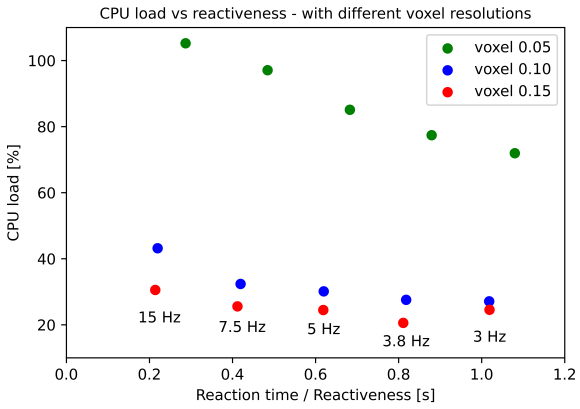


Figure 11: CPU load for various reaction times with voxel resolutions 0.05 m, 0.10 m, and 0.15 m.

http://www.imavs.org/

http://www.imavs.org/

5.5 *Trade-off: Safety vs. Performance*

By combining the results, trade-offs can be visualised in which the CPU loads can be plotted for a combination of two performance variations, also known as a Pareto-front [22]. The main objective is the comparison of CPU load relative to flight performance, which is represented by the minimal gap size and maximum obstruction width. Safe and robust flight was one of the secondary objectives, and this is represented by reactiveness and minimal obstacle-size performance. Exploring the trade-off between safety and performance at different CPU utilization, using reactiveness and minimal gap size, reveals a Pareto-front.

The Pareto-fronts, shown in Figure 12, indicate the performance combinations where the CPU load is equal, drawn in yellow for 90% CPU utilisation, and drawn in purple for 50% CPU utilisation. This Pareto-front shows the trade-off where a combination of safety and performance has to be chosen, with a limited CPU load, that is most suitable for a design situation.

For example, in the trade-off in Figure 12, for a CPU load limit of 50%, a minimum gap size of 0.8 m can be chosen with a reactiveness of 1.2 seconds. Or, the combination with a much quicker reactiveness can be chosen, for example, 0.2 seconds, but now the minimum gap size is 1.1 m. Any combination on the purple line would be valid within the 50% load limit.



Figure 12: Trade-off between safety and performance shown as Pareto fronts for CPU utilization levels of 50% and 90%. Each curve illustrates feasible combinations of reactiveness and minimum gap size under processor constraints.

6 Autonomous Flight with a 100-Gram Drone

Unlike prior sub-100 g autonomous systems, which are limited to route following in controlled indoor environments [10], the platform presented here performs real-time 3D obstacle mapping and path planning in unknown, unstructured environments. This is achieved by fitting a Raspberry Pi Zero 2W processor and an Arducam ToF camera onto a DJI Tello drone, resulting in a fully autonomous 100-gram, 10×10×10 cm system with all computation performed onboard. The total material cost is approximately 200 euros, of which 100 euros is the drone platform itself, making it a low-cost solution compared to existing autonomous systems of comparable capability. Low-level flight stabilization and velocity control are handled by the Tello's internal controller, with high-level commands sent via the DJITelloPy SDK [23].

The parameter configuration for the 100-gram platform was derived directly from the benchmarking results presented in Section 5, using real forest sensor data as input. By selecting a 10 cm voxel grid resolution, 10 depth images per second at 120×90 px resolution, and a 5 m local planner range, the processor operates at 80–95% CPU load — within the safe limits of stable operation on the Raspberry Pi Zero 2W. This configuration yields the following navigation performance: a minimum solvable gap size of 1.0 m, a reaction time of 300 ms, a minimum detectable obstacle size of 2 cm, and a maximum solvable obstruction width of 3.6 m. These values represent a deliberate trade-off between computational feasibility and navigation safety, selected using the Pareto-front analysis described in Section 5.

Preliminary flight tests were conducted in a controlled indoor environment with artificial vegetation arranged at approximately 1.0 m gap spacing, consistent with the predicted minimum solvable gap size. The drone successfully navigated autonomously between obstacles, confirming the benchmarking-derived parameter configuration in practice. Outdoor flights in a forest environment demonstrated functional obstacle avoidance, though thin branches were occasionally detected only at close range, consistent with the 2 cm minimum detectable obstacle size and 300 ms reaction time reported above. The addition of the companion computer and sensor payload reduced flight time from the platform's original 13 minutes to approximately 6–7 minutes. Full quantitative outdoor validation remains future work.

7 Conclusion

This study demonstrates that downsizing hardware and adapting algorithms enable small drones to achieve effective autonomous navigation in dense rainforest conditions. By systematically varying parameters such as voxel grid resolution, camera resolution, and local planning range, this research shows clear trade-offs between processor load, memory usage, and crucial performance metrics like minimal gap size, obstacle detection, and reaction time. Our findings indicate that reductions in image rate or resolution can substantially decrease CPU load but must be carefully balanced against the increased risk of missing small obstacles, or reacting too slowly to new hazards. This balance is further illustrated through Pareto-front analyses, offering insights into optimal configurations for different mission requirements. Fi-

nally, the successful implementation of autonomous 3D obstacle mapping and path planning on a 100-gram drone platform confirms that real-time autonomous navigation through unknown, unstructured environments is feasible at this scale. This was demonstrated in both controlled indoor environments and outdoor forest conditions. These results open a promising path for future applications in biodiversity monitoring and ecological conservation.

ACKNOWLEDGEMENTS

The authors would like to thank Erik van der Horst for his support at the lab and during outdoor flights.

REFERENCES

[1] Stephen Pringle, Martin Dallimer, et al. Opportunities and challenges for monitoring terrestrial biodiversity in the robotics age. *Nature Ecology & Evolution*, pages 1–12, 2025.

[2] Salua Hamaza, H Nguyen, et al. Sensor delivery in forests with aerial robots: A new paradigm for environmental monitoring. In *IEEE IROS Workshop on Perception, Planning and Mobility in Forestry Robotics*, 2020.

[3] Liming Zheng and Salua Hamaza. Albero: Agile landing on branches for environmental robotics operations. *IEEE Robotics and Automation Letters*, 9(3):2845–2852, 2024.

[4] Emanuele Aucone, Steffen Kirchgeorg, et al. Drone-assisted collection of environmental DNA from tree branches for biodiversity monitoring. *Science Robotics*, 8(74):eadd5762, 2023.

[5] Christian Geckeler, Steffen Kirchgeorg, et al. Field deployment of BiodivX drones in the Amazon rainforest for biodiversity monitoring. *IEEE Transactions on Field Robotics*, 2025.

[6] Rita Santos Raminhos and Salua Hamaza. Task-oriented path planning towards autonomous sensor network deployment in rainforest canopies. *Advanced Robotics Research*, 2026.

[7] Yulun Tian, Katherine Liu, et al. Under canopy search and rescue using multiple UAVs. *International Journal of Robotics Research*, 39(10-11):1201–1221, 2020.

[8] Philipp Foehn, Elia Kaufmann, et al. Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight. *Science Robotics*, 7(67):eabl6259, 2022.

[9] Xin Zhou, Zhepei Wang, et al. Ego-planner: An ESDF-free gradient-based local planner for quadrotors. *IEEE Robotics and Automation Letters*, 6(2):478–485, 2020.

[10] Tom van Dijk, Christophe De Wagter, and Guido C.H.E. de Croon. Visual route following for tiny autonomous robots. *Science Robotics*, 9(92), 2024.

[11] XPRIZE. XPRIZE rainforest—XPRIZE foundation. `https://www.xprize.org/prizes/rainforest`, 2023.

[12] Mani Monajjemi. bebop_autonomy: ROS driver for Parrot Bebop drone. `https://bebop-autonomy.readthedocs.io/en/latest/`, 2014.

[13] Ma Weinmann, Mi Weinmann, et al. Fast and automatic image-based registration of TLS data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(6 SUPPL.), 12 2011.

[14] Sebastian Thrun, Wolfram Burgard, et al. *Probabilistic Robotics*. Intelligent Robotics and Autonomous Agents. MIT Press, Cambridge, MA, 2005.

[15] Armin Hornung, Kai M. Wurm, et al. Octomap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013.

[16] Alexander Spitz. cpu_monitor: CPU and memory monitoring ROS package. `https://github.com/alspitz/cpu_monitor`, 2019.

[17] Tong Qin, Shaozu Cao, et al. A general optimization-based framework for global pose estimation with multiple sensors. *arXiv preprint*, 2019.

[18] 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.

[19] 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.

[20] Eric Hyyppa, Juha Hyyppa, et al. Under-canopy UAV laser scanning for accurate forest field measurements. *ISPRS Journal of Photogrammetry and Remote Sensing*, 164:41–60, 6 2020.

[21] Vaino Karjalainen, Niko Koivumaki, et al. Autonomous under-canopy forest mapping using stereo vision and photogrammetric reconstruction. *International Journal of Remote Sensing*, 2025.

[22] Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001.

[23] Damia Fuentes. DJITelloPy: DJI tello drone Python interface. `https://github.com/damiafuentes/DJITelloPy`, 2021.

http://www.imavs.org/

Safe synchronous path planning for a fleet of fixed-wing constant speed aircraft

Maël Feurgard, Gautier Hattenberger, Nicolas Durand[1,*] and Simon Lacroix[2,†]
Fédération ENAC ISAE-SUPAERO ONERA, Université de Toulouse, France[1]
RIS, Laboratoire d'Analyse et d'Architecture des Systèmes, Université de Toulouse, CNRS, Toulouse, France[2]

http://www.imavs.org/

Abstract

Path planning for multiple unmanned aerial vehicles is a difficult task, and even more for a fleet of fixed-wing aircraft. One specific case is the transition to, or between, formation flight patterns, which requires synchronous arrivals while ensuring minimal separation, and ideally maintaining cruise speed. We present a centralized method to solve this problem based on enumerating different Dubins paths. Given a travel time for the fleet, it builds a set of possible paths for each aircraft. Then, it checks in parallel separation between each path pair. This yields coefficients for an Integer Linear Programming problem determining if a fleet-wide conflict-free solution exists. This process is repeated for different travel times sampled with increasing resolution until the user-defined accuracy is met. The method is benchmarked with Monte-Carlo simulations considering up to 20 aircraft simultaneously, achieving an 95% success rate for an average 8 seconds computation time. A showcase integration is done using three simulated aircraft within the PaparazziUAV simulator.

1 Introduction

Unmanned Aerial Vehicles (UAVs) are becoming increasingly affordable at relatively low prices, thereby encouraging the deployment of multiple autonomous aircraft simultaneously. In particular, a fleet of fixed-wing aircraft is useful for long missions, requiring significant payload or airspeed ; aerial cartography [1], airborne particles sensing [2] and forest monitoring [3] are examples of such tasks. They all require solving a common problem: cooperative path planning.

Cooperative path planning consists in moving multiple agents in a shared space to achieve a common goal. Several constraints can be considered, the most common being collision avoidance, synchronous arrival to destination, maintaining formation and feasible dynamics. These problems have been significantly studied for aerial vehicles [4, 5, 6]. We are particularly interested in collision-free 2D movements with synchronous arrivals, since such movements correspond to getting into a planar formation or switching between formations. Different strategies exist to achieve such transitions: discretize the airspace using a grid[7]or sets of lines [8] first, then generate safe flight paths and finally smooth them; use leader-follower or swarm schemes to handle collision avoidance dynamically, and vary speed for synchronization [9, 10, 11]; or directly generate valid paths.

We focus on this last category as it does not rely on a discretization parameter and provide an explicit path to be followed for each agent. This ensures that any solution is deadlock-free, but it usually requires more computing power and there is no guarantee that a solution will be found. Paths can be generated along several models. Some can consider the acceleration limitations of the aircraft, and will use clothoids [12], Bézier splines [13, 14] or Pythagorean Hodograph splines [15, 16, 17]. Others use the Dubins model [18], as it simplifies trajectory generation into a geometry problem with circles and lines [19, 20] at the cost of introducing acceleration discontinuities.

Among the full path generation methods, only a limited amount of aircraft are considered and performances are poorly studied. In [14], several Monte Carlo simulations are done, measuring accuracy and runtime of the planner, testing from 2 to 7 aircraft. But this article does not manage collisions. The other articles show only sample cases, without timing computations and the number of aircraft does not exceed 5. This is especially limited compared to the leader follower techniques, such as in [9] where 3 groups of 7 aircraft were deployed in real conditions.

We present a new method based on *Dubins paths* to solve the fixed-wing fleet path planning problem with the constraints of constant airspeed, collision avoidance and synchronous arrival in 2D. The main difference with [19, 20] is that instead of looking for one specific combination of curves in a continuous domain satisfying both synchronization and collision avoidance, we build sets of synchronous curves for each aircraft in which we look for a collision-free combination. This allows us to parallelize computations and push the benchmark from 3 to 20 aircraft simultaneously.

2 Problem formulation

We consider the problem of path planning for a fleet of fixed-wing aircraft flying at constant air speed and altitude, using simplified dynamics in an obstacle-free area.

*Email adresses: [name].[surname]@enac.fr
†Deceased, https://sites.laas.fr/Simon_Lacroix/

http://www.imavs.org/

Let the aircraft be numbered from 1 to N, and represented using a 2D position $[x_k, y_k] \in \mathbb{R}^2$ and a 2D orientation $\theta_k \in \mathbb{S} := (-\pi, \pi]$ with respect to the ground. We combine these values in a state vector $\mathbf{p}_k = [x_k, y_k, \theta_k]$. For the sake of simplicity, we assume all aircraft have the same characteristics, namely the same cruise air speed V, minimal turn radius $\rho_{\min}$, and minimal separation distance δ.

This yields the following dynamics for each aircraft $k \in [1, N]$, the so-called *Dubins vehicle* [18]:

$$\frac{\mathrm{d}\mathbf{p}_k}{\mathrm{d}t} = \begin{pmatrix} \dot{x_k} \\ \dot{y_k} \\ \dot{\theta_k} \end{pmatrix} := \begin{pmatrix} V\cos\theta_k \\ V\sin\theta_k \\ u_k \end{pmatrix} \tag{1}$$

The sole control parameter is the change in orientation $u_k \in [-V/\rho_{\min}, V/\rho_{\min}]$. Note that the vehicle is always aligned with its speed vector. As such, for a differentiable trajectory $\gamma : \mathbb{R} \to \mathbb{R}^2$, we may abuse the notation $\mathbf{p}_k(t) = \gamma(t)$ to state that $\gamma(t) = [x_k(t), y_k(t)]$ and $\gamma'(t)$ is aligned with $[\cos\theta_k(t), \sin\theta_k(t)]$, i.e. the aircraft follows γ both in position and orientation.

We require synchronicity: all aircraft reach their destination at the same time. This constraint stems from formation flights, and can be adapted to sequencing for instance. We develop this aspect in section 4.

Let us formulate our optimization problem. Given for each aircraft k an initial state $\mathbf{S}_k$, a final state $\mathbf{E}_k$ both in $\mathbb{R}^2 \times \mathbb{S}$, and a set of possible twice differentiable paths $\Gamma \subset \mathcal{D}^2(\mathbb{R}^+, \mathbb{R}^2)$, we seek to minimize the time taken by the fleet to reach its final configuration while satisfying the constraints. Formally:

$$\min_{\substack{\tau \in \mathbb{R}^+ \\ \gamma_k \in \Gamma, 1 \le k \le N}} \quad \tau \qquad \text{(Flight time)} \tag{2}$$

Such that we achieve:

1. Correct synchronous trajectory:

$$\forall\, k \in [1, N] \quad \gamma_k(0) = \mathbf{S}_k \ \wedge\ \gamma_k(\tau) = \mathbf{E}_k$$

2. Collision avoidance:

$$\forall\, k, s,\ k \neq s \quad \forall\, t \in [0, \tau]\ \ \|\gamma_k(t) - \gamma_s(t)\| > \delta$$

3. Constant speed:

$$\forall\, k \quad \forall\, t \in [0, \tau]\ \ \|\gamma_k'(t)\| = V$$

4. Bounded acceleration:

$$\forall\, k \quad \forall\, t \in [0, \tau]\ \ \|\gamma_k''(t)\| \le \frac{V^2}{\rho_{\min}}$$

The core starting point for solving this problem is to define the set of possible paths Γ. This set is often made of splines [13, 15, 16, 17] as they provide a continuous family with a convenient representation for integrating spatial and dynamic constraints in a solver. But they heavily rely on a good initial guess and continuous optimization. This can be tempered by the use of global heuristics like Particle Swarm Optimization (PSO), at a significant computational cost.

The method presented here is based on a finite set of solutions. It has the advantage of exploring significantly different paths, which may then be used as a better starting point for locally optimized methods. We now present the elements making our set of solutions: the *Dubins paths*.

3 Dubins paths

Dubins showed in [18] that the shortest path between any two states of a vehicle described in (1) is made of straight lines and circle arcs of radius $\rho_{\min}$. We denote the possible components S, L, R, respectively for Straight, Left and Right [turn]. The optimal solution is among the following six possibilities: RSR, LSL ; RSL, LSR and LRL, RLR.

We may still need more variety in the choice of paths in order to avoid conflicts. We add the single-turn paths SLS and SRS. They have a limited scope of application, but can still extend the number of possibilities. With the six basic Dubins paths and these two, we define the set of *basic paths* γ_0, which has cardinality 8. For any type $\gamma \in \gamma_0$, a path is exactly defined by starting and ending poses $\mathbf{S}, \mathbf{E} \in \mathbb{R}^2 \times \mathbb{S}$ and a turn radius $\rho > 0$, and we denote $\gamma[\mathbf{S}, \mathbf{E}, \rho] \in \mathcal{D}^2(\mathbb{R}, \mathbb{R}^2)$ the corresponding function, when it exists. They are all showcased in Figure 1.

By construction, all of these solutions are guaranteed to satisfy the Constant speed and Bounded acceleration constraints. Two main questions remain: how to detect conflicts and how to tune the paths to achieve synchronicity ?

3.1 *Finding a fitting solution*

Dubins paths provide a direct solution to the individual path planning problem, but offer little tuning mechanisms for achieving a specific path length. This question has been studied, and several methods devised: artificially increase the turning radius [19], use circles with different radii [21], in-

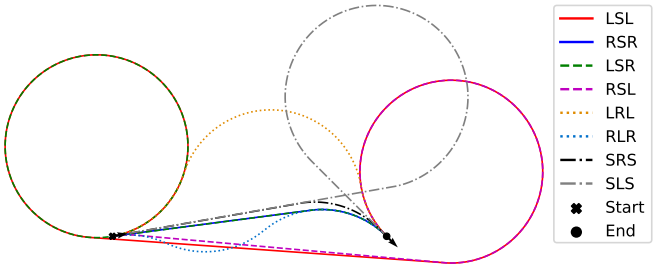


Figure 1: Examples of the six Dubins paths plus two extras (SRS, SLS)

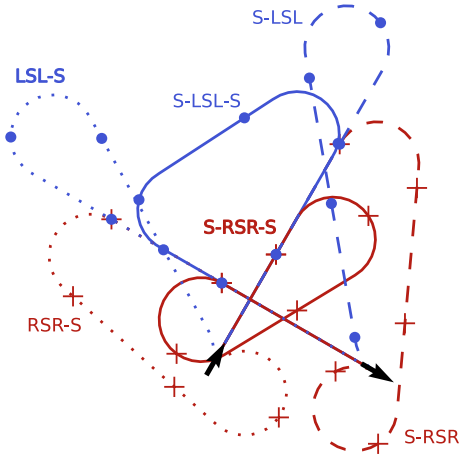


Figure 2: Start, end and both extended paths: RSR and LSL

troduce straight extensions [22] or deform straight parts [23]. We use two strategies: increasing the turn radius and adding straight segments at the start or end of the path.

More formally, fitting length $\ell > 0$ by tweaking radius ρ of a path from $\mathbf{S} = [x_S, y_S, \theta_S]$ to $\mathbf{E} = [x_E, y_E, \theta_E]$ corresponds to the following optimization problem:

$$\min_{\rho > \rho_{\min}} (\text{len}(\gamma[\mathbf{S}, \mathbf{E}, \rho]) - \ell)^2 \tag{3}$$

Similarly, using length extensions distributed with ratio $r \in [0, 1]$ along the initial direction $\mathbf{V_S} = [\cos\theta_S, \sin\theta_S, 0]$ and final one $\mathbf{V_E} = [\cos\theta_E, \sin\theta_E, 0]$ yields the problem:

$$\min_{l \geq 0} (\text{len}(\gamma[\mathbf{S} + lr\mathbf{V_S}, \mathbf{E} - l(1-r)\mathbf{V_E}, \rho_{\min}]) + l - \ell)^2 \tag{4}$$

Note that the second method introduces new types of paths: Dubins extended by straight lines. The ratio parameter allows many variations, but for the sake of simplicity we will only keep three: start extended ($r = 1$), noted S-*AAA*; end extended ($r = 0$) *AAA*-S and both extended ($r = 1/2$) S-*AAA*-S, where in all cases *AAA* represents one of the three letters combination characterizing an element of γ_0. A showcase of some Dubins extended paths is given in Figure 2. Let Γ define the set containing the *basic paths* γ_0 length fitted using the aforementioned methods; its cardinal is $8 + 3 \cdot 8$.

Solving problems (3) and (4) is manageable since they are unidimensional and quite monotonous, in spite of some discontinuities (see [14, Figure 4] for path length as a function of the turn radius). We use Brent's method [24, Chapter 4] to efficiently find solutions.

3.2 *Conflicts detection*

Dubins path are made of straight lines and circle arcs. Thus the question of detecting a conflict between two paths of Γ can be broken down into finding conflicts between their components. Three subcases emerge: line-line, line-circle and circle-circle. Both 2D lines and circles can be conveniently represented using complex functions.

For two parametric functions $f_1, f_2 : [0, \tau] \to \mathbb{C}$ representing two trajectories, we define two types of separations with respect to some minimal distance $\delta > 0$:

- *Temporal separation* : two points following prescribed trajectories must stay separated by at least δ.

$$\min_{t \in [0,\tau]} |f_1(t) - f_2(t)| > \delta \tag{5}$$

- *Spatial separation* : two shapes must be separated by at least δ.

$$\min_{(t_1,t_2) \in [0,\tau]^2} |f_1(t_1) - f_2(t_2)| > \delta \tag{6}$$

One can easily verify that if (6) is satisfied, then so is (5).

Practically, we are interested in *temporal separation*. It is easy to compute in the line-line case, as it becomes a second degree polynomial equation. But there are no such simplifications for the others ; we require a global solver, the interval method [25] for instance. To limit the use of such computationally expensive procedures, we can perform pre-filtering by computing the *spatial separation*, which is straightforward given the geometric nature of the paths considered.

We have presented in this section the set of paths Γ used for solving problem (2). We have shown how to fit them to specific length and how to detect conflicts between them. Let us now put this together into a fleet path planner.

4 FLEET PATH PLANNING

4.1 *Presentation of the algorithm*

Our procedure for planning a path for the fleet respecting the constraints given in (2) uses combinatorial optimization. We consider two main factors regarding complexity: the number of different possible paths $|\Gamma|$ and the number of aircraft N. The former can be increased or reduced by modifying the chosen individual path planning methods, and the latter expresses the scale and difficulty of the problem.

Three primitives are required to describe our method, whose underlying principles have been given in section 3:

- $\tau_k, \gamma_k \longleftarrow \text{ShortestDubins}(\mathbf{S}_k, \mathbf{E}_k, V, \rho_{\min})$: Given starting and ending states $\mathbf{S}_k, \mathbf{E}_k$, the aircraft speed V and its minimal turn radius $\rho_{\min}$, compute the shortest path γ_k belonging to Γ and give the travel time τ_k.

  Its complexity scales linearly with $|\Gamma|$, since each path must be considered to find the shortest.

- $L_k \longleftarrow \text{FitDubins}(\mathbf{S}_k, \mathbf{E}_k, V, \rho_{\min}, \tau)$: Given starting and ending states $\mathbf{S}_k, \mathbf{E}_k$, the aircraft speed V, its minimal turn radius $\rho_{\min}$ and a travel time τ, output a set of paths L_k taken from Γ with turning radii greater or equal than $\rho_{\min}$ and having a travel time of τ. This set may be empty if there is no way to go from $\mathbf{S}_k$ to $\mathbf{E}_k$ given the constraints and possible paths.

  Its complexity also scales linearly with $|\Gamma|$, since each path must be considered.

http://www.imavs.org/

- True|False $\longleftarrow$ AreSeparated$(\delta, \gamma_1, \ldots, \gamma_N)$: Given a minimal separation distance δ and a list of paths $\gamma_1, \ldots, \gamma_N$, check if there is a loss of separation between any two pairs.

  We can safely assume that there is an underlying IsPairSeparated$(\delta, \gamma_a, \gamma_b)$ subroutine checking the separation between two given paths, solving *spatial* (6) then possibly *temporal* (5) separations. AreSeparated requires $\frac{N(N-1)}{2} = \mathcal{O}\left(N^2\right)$ calls to the subroutine (since separation is a symmetric notion).

We can now describe the general principles, also summarized in pseudocode algorithm 1. It starts by computing for each aircraft its best individual solution, ignoring collisions, by using the ShortestDubins primitive. We want synchronous arrivals, i.e. same travel time for everyone. A first good guess is the maximum of the minimum individual times, which we denote $\tau_{\min}$. Using a user-defined parameter R, the maximum possible time is set to $R \cdot \tau_{\min}$. We use these two values to initialize a priority queue Q containing the flight times considered.

The rest of algorithm 1 executes the main loop until a stop condition is met: For every untested flight time τ in Q, compute the possible paths L_k for each aircraft $k \in [1, N]$ using FitDubins, and look through all combinations $(\gamma_1, \ldots \gamma_N) \in L_1 \times \cdots \times L_N$; if one satisfies AreSeparated with the desired separation δ, break the loop, otherwise move to the next value in Q.

If a solution is found, remove all elements in Q with flight time greater than the one of the solution, since they cannot be optimal. Then, Q is populated by inserting b values regularly spaced between every two successive $\tau_i, \tau_{i+1} \in Q$ (if $b = 1$, this is equivalent to successive dichotomies). To limit sampling to a reasonable resolution, a minimum width $w > 0$ is specified by the user, such that if $\tau_{i+1} - \tau_i \leq w$, no samples are added between these two values.

The number of iterations is counted as the number of different flight times τ tested. Once above a given threshold, the program stops. The second stop condition is the maximum total execution time, and is evaluated before testing a new τ. The last stop condition is a progress criterion: the program stops if no new τ are added to Q when re-sampling.

4.2 Algorithm analysis

The different stop conditions ensure that the program terminates in a timely manner, but does not guarantee that a solution is found. Furthermore, there is no guarantee at large to find a solution, even with unlimited computations, as it may not exist in the Γ set chosen. Still, we can evaluate the complexity of testing a flight duration τ, which is the core of this algorithm. Computing the set of possible paths for each aircraft contributes $N \cdot \mathcal{O}(|\Gamma|)$. Then, there are at most $|\Gamma|$ paths per aircraft, meaning a total of $|\Gamma|^N$ combinations. Each combination has to be tested for conflicts, yielding a

Algorithm 1: Dubins based path planning algorithm

Data: Initial and final states $(\mathbf{S}_k, \mathbf{E}_k)_{1 \leq k \leq N}$; flight parameters $V, \rho_{\min}, \delta$; optimization parameters R, w, b
Result: Duration τ and Dubins trajectories $(\gamma_k)_{1 \leq k \leq N}$

```
for 1 ≤ k ≤ N do    // Initial solutions
    τ_k, γ_k ⟵ ShortestDubins(S_k, E_k, V, ρ_min)
end
τ_min ⟵ max_{1≤k≤N}(τ_k)
// Ascending priority queue
Q ⟵ [τ_min; R · τ_min]
Best ⟵ ∅
repeat
    foreach untested τ ∈ Q do
        // Compute all solutions for a given length
        for 1 ≤ k ≤ N do
            L_k ⟵ FitDubins(S_k, E_k, V, ρ_min, τ)
        end
        // If one L_k is empty, this loop is skipped
        foreach Combination (γ_1, ..., γ_N) of L_1 × ··· × L_N do
            if AreSeparated(δ, γ_1, ..., γ_N) then
                Best ⟵ τ, (γ_1, ..., γ_N)
                break twice
            end
        end
        Mark τ as tested
    end
    if Best is not empty then
        Remove from Q all τ >Best[0]
    end
    for Each successive pair τ_i, τ_{i+1} ∈ Q do
        Δτ ⟵ τ_{i+1} − τ_i
        if Δτ > w then
            for 1 ≤ j ≤ b do Add τ_i + j Δτ/(b+1) to Q
        end
    end
until Stop condition
return Best
```

http://www.imavs.org/

cost per iteration of $\mathcal{O}\left(N^2 |\Gamma|^N\right)$. This is *polynomial* with respect to the number of different paths $|\Gamma|$ and *exponential* with respect to the number of aircraft N.

Here are two ways we used in our implementation to speed up computations. Since conflicts involve aircraft two by two, it is possible to compute for each of the $\frac{N(N-1)}{2}$ unordered aircraft pairs the $|\Gamma|^2$ paths pairs, and check for conflict. This provides $N^2 |\Gamma|^2$ binary coefficients $c_{a,b,g,h}$ denoting whether there is a conflict between aircraft a and b using respectively paths g and h. We can formulate a *0-1 Integer Linear Program* (ILP) with variables $x_{a,g}$ defining if aircraft a uses path g. There are two types of constraints: avoiding conflicts

$$\forall a \neq b \in [1, N] \; \forall g, h \in \Gamma \; c_{a,b,g,h} \cdot (x_{a,g} + x_{b,h}) \leq 1$$

and choosing a unique path for each aircraft

$$\forall a \in [1, N] \; \sum_{g \in \Gamma} x_{a,g} = 1$$

This is a well-known NP-complete problem, and an efficient existing solver can be used, for instance the HiGHS solver [26].

Another important point is that this algorithm is easily parallelizable. Indeed, each aircraft k can independently compute its set of possible paths L_k, and each conflict check is independent of the others. Hence, if we consider a centralized multiprocessor unit, we may use at most $\frac{|\Gamma|^2 N(N-1)}{2}$ cores to compute the aforementioned $c_{a,b,g,h}$.

4.3 Algorithm extensions

The capabilities of algorithm 1 can be extended by a clever use of its formulation. The first one is that it is possible to take into account a constant uniform wind W. Since the flight time τ is set before planning the paths, it is possible to update the ending point by integrating the wind deviation. This amounts to replace $\mathbf{E}_k$ by $\mathbf{E}_k - W\tau$, thereby anticipating the wind induced drift. Note that since the same linear change of referential is applied to every aircraft, conflict detection does not need to be changed. Regarding the initial guess for $\tau_{\min}$, there exist methods to compute it in presence of uniform wind, for instance [27].

The second modification consists in incorporating temporal separation at arrival. Assume that aircraft $k \in [2, N]$ must reach their destination with a time difference Δt_k with respect to the previous aircraft $k-1$. This is implemented by replacing τ in the call of FitDubins by $\tau + \sum_{j=1}^{k-1} \Delta t_j$. One use case of such modification is landing sequencing.

5 Simulations

The method is tested in simulation with 3 types of scenarios: transition between formations, random state to formation and random state to another random state, respectively referred to as Formation, RNG to Formation and Full RNG in

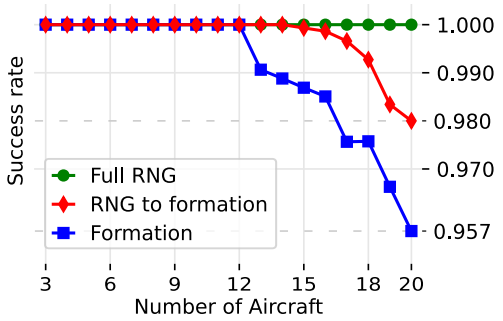


Figure 3: Success rate as function of aircraft number and type of test case

Figure 3. A total of 22 formations were considered, ranging from circular and linear to rectangular and their variants.

We based our simulations on a fixed-wing drone of $1\,\mathrm{m}$ wingspan (equivalent to a Parrot Disco). The airspeed is set to $15\,\mathrm{m/s}$, minimal turn radius to $40\,\mathrm{m}$ and minimal separation to $80\,\mathrm{m}$. When using random positions, sampling was done in a $1 \times 1\,\mathrm{km}$ area with $240\,\mathrm{m}$ of distance requested. When using positions in formation, this distance between reference positions is set to $120\,\mathrm{m}$. The distance between initial and final formations is of $1\,\mathrm{km}$. No wind was considered in these tests. Regarding the algorithm parameters, we set $R = 10$, $b = 2$ and $w = \max(0.1\,\mathrm{s}, R \cdot \tau_{\min} \cdot 10^{-4})$. The stop conditions values are 300 iterations maximum and a $60\,\mathrm{s}$ timeout. The solver has been benchmarked on a 12th Gen Intel(R) Core(TM) i7-1255U, using 12 threads.

We ran tests with fleets of 3 to 20 aircraft, with 5230 cases each: 300 fully random, 530 formation transitions and 4400 random to formation. The proportion of successful cases per aircraft number and problem type is reported in Figure 3. Success rate starts to decrease for more than 12 aircraft. Two main reasons can be given for failures. First, formation cases use tighter margins at endpoints and do not optimize aircraft reordering between formations. Instead, they use an arbitrary order, which may be unnecessarily difficult. Second, randomly selected initial points may hinder synchronicity, especially when the arrivals are tightly ordered. Nevertheless, for 12 aircraft and less, no failures were reported, and the rate stays above 95% when testing with 20 aircraft.

Computation times are reported in Figure 4. We measured the main thread time since parallelism has been exploited in our implementation. For 9 aircraft and below, 99% of cases are solved in less than $1\,\mathrm{s}$. For 20 aircraft, 90% of cases were completed in less than $13\,\mathrm{s}$. A detailed analysis reveals that the $60\,\mathrm{s}$ timeout is reached only for some cases with 17 aircraft or more. The number of iterations is never a limiting factor; the maximal value measured is 87 iterations.

The most difficult cases are the transitions between formations, then random to formations and finally fully random; the computation times with 20 aircraft average to $5065\,\mathrm{ms}$, $3877\,\mathrm{ms}$ and $417\,\mathrm{ms}$ respectively. It can be explained by the

http://www.imavs.org/

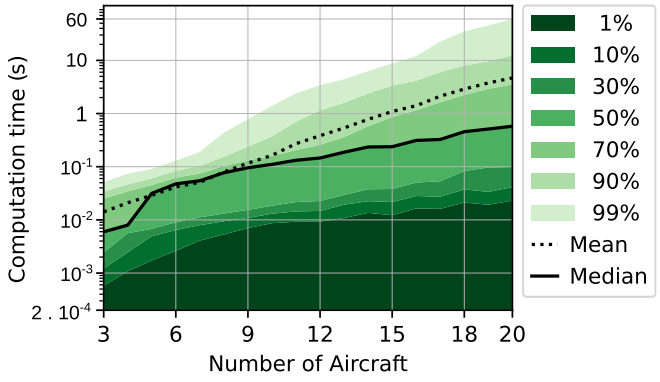


Figure 4: Computation time log distribution as function of aircraft number

lack of geometric separation between trajectories with formations, as it incurs the use of more time-consuming algorithms for detecting potential conflicts.

Preliminary integration of the method has been tested using the 6 degrees of freedom simulator from PaparazziUAV [28]. A centralized planner gather data from three simulated aircraft sharing the same characteristics: nominal airspeed set to $15\,\mathrm{m/s}$, minimal turn radius tuned to $40\,\mathrm{m}$ and minimal separation to $30\,\mathrm{m}$. This last parameter is significantly smaller compared to our initial tests in order to keep the planned trajectories in a reasonable airspace. Dynamic replanning is set up with the following rules, by decreasing order of priority: a new plan can be established at most once every $15\,\mathrm{s}$, the current plan is considered invalid if a deviation of $10\,\mathrm{m}$ is measured for one aircraft, a new plan can be proposed if it reduces flight time by 10%, the current plan ends at $8\,\mathrm{s}$ from arrival. Tracking is achieved using the predefined line and circle primitives of the autopilot. Each aircraft adjusts its own airspeed to try reducing tracking error. It is based on the time-to-arrival error, with adjustments within $\pm 1\,\mathrm{m/s}$ of its cruise speed.

One of the simulated flights is showcased in Figure 5. The objective is to achieve four straights in chevron formation with a spacing of $36\,\mathrm{m}$. The highlighted section contains the transition with the largest error. This error for aircraft 3 can be attributed to cutting the start of its turn and failing to slow down efficiently. This deviation triggered at $161\,\mathrm{s}$ a rescheduling, which at first increased the tracking error due to the sudden change in command, followed by a decrease as control stabilized. More generally, these simulations show a satisfying planning strategy, especially considering the naive tracking implemented. Still, the tracking error is not negligible and some margins have to be considered between the separation requested to the solver and the actual minimal tolerable separation.

6 Conclusions and Future Works

In this article we showed a method to solve synchronous path planning for a fleet of constant speed Dubins vehicles, and benchmarked it from 3 to 20 aircraft. On average, even up to 20 aircraft, the computation time is $8\,\mathrm{s}$ for a success rate superior to 95%. The method enforces synchronicity by setting the travel time first, then creating a finite set of paths per aircraft satisfying the constraints, and finally seeking a conflict-free combination, which is a solution to the fleet path planning problem. Simulated flights within the Paparazzi-UAV system show correct tracking given aircraft dynamics, but highlight the possible improvements in plan execution.

This algorithm has several limitations, the first one being that there are no guarantees to find a solution. This may be mitigated by adding possible paths and reducing margins, but these come with computational and safety costs. Secondly, computational cost scales exponentially with the number of aircraft, limiting scalability. This method was not made to replace swarm-like tactics used to handle large amounts of UAVs, but the time to find a solution can be an obstacle depending on the available hardware and expected reactivity. Lastly, the paths presented here are based on Dubins model. If it simplifies greatly path planning, it does not match an actual aircraft model, since lateral acceleration is discontinuous. Given how important separation between aircraft is, we wish that path following is as precise as possible.

There are multiple ways of improving this method. The most promising direction is to locally post-process the generated paths using B-Splines [14] or Pythagorean Hodograph splines [15, 17] in order to fix discontinuities, and then improve tracking accuracy and avoid collisions using appropriate guidance techniques, such as Guiding Vector Fields [29]. One could also improve computation time by studying how conflicts between paths evolve as they are continuously deformed to fit a different length. Lastly, other extensions are possible, such as integrating vertical movement [30].

Our C++ implementation of the solver can be found at https://github.com/enacuavlab/DubinsFleetPlanner.

References

[1] Esther Salamí, Cristina Barrado, and Enric Pastor. Uav flight experiments applied to the remote sensing of vegetated areas. *Remote Sensing*, 6(11):11051–11081, November 2014. ISSN: 2072-4292. DOI: 10.3390/rs61111051.

[2] Felipe Gonzalez, Marcos P.G. Castro, Pritesh Narayan, Rod Walker, and Les Zeller. Development of an autonomous unmanned aerial system to collect time-stamped samples from the atmosphere and localize potential pathogen sources. *Journal of Field Robotics*, 28(6):961–976, October 2011. ISSN: 1556-4967. DOI: 10.1002/rob.20417.

[3] Lucio R. Salinas, Georgios Tzoumas, Lenka Pitonakova, and Sabine Hauert. Digital twin technology for wildfire monitoring using uav swarms. In *2023 International Conference on Unmanned Aircraft Systems*

http://www.imavs.org/

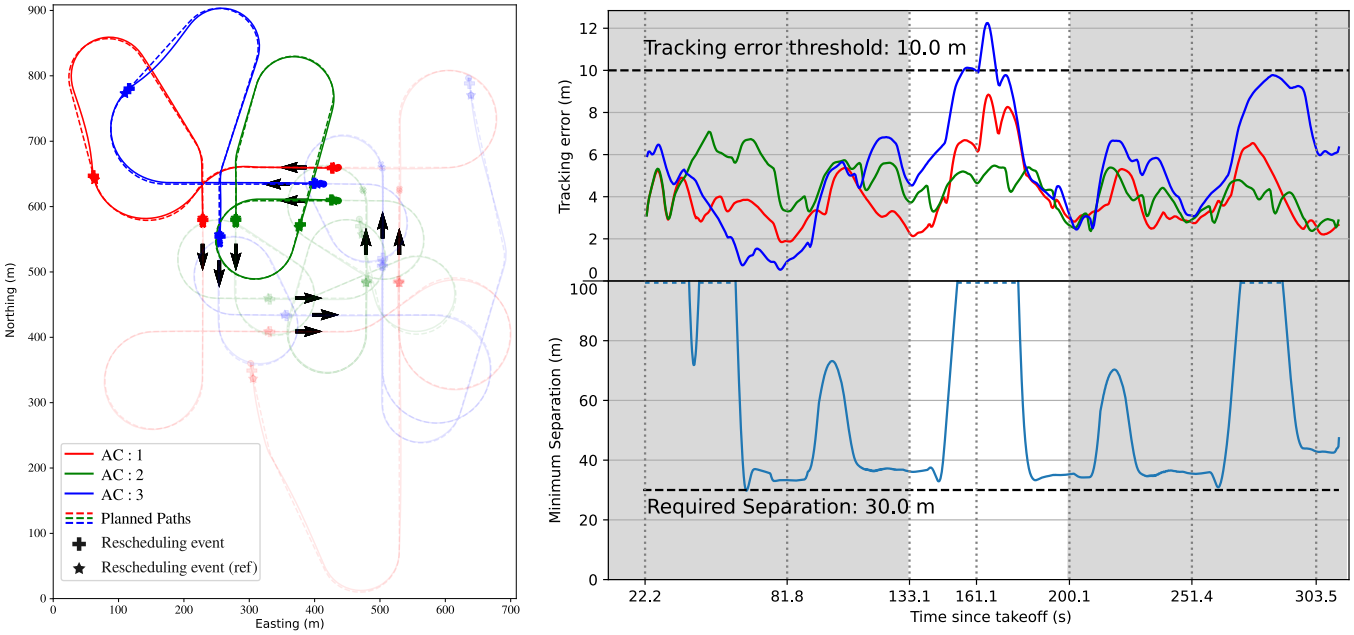


Figure 5: Flight of three identical aircraft simulated using PaparazziUAV. Separation measured up to 100 m

(ICUAS), pages 586–593. IEEE, June 2023. DOI: 10.1109/icuas57906.2023.10155819.

[4] Hao Zhang, Bin Xin, Li-hua Dou, Jie Chen, and Kaoru Hirota. A review of cooperative path planning of an unmanned aerial vehicle group. *Frontiers of Information Technology & Electronic Engineering*, 21(12):1671–1694, December 2020. ISSN: 2095-9230. DOI: 10.1631/fitee.2000228.

[5] Soon-Jo Chung, Aditya Avinash Paranjape, Philip Dames, Shaojie Shen, and Vijay Kumar. A survey on aerial swarm robotics. *IEEE Transactions on Robotics*, 34(4):837–855, 4, August 2018. ISSN: 1941-0468. DOI: 10.1109/tro.2018.2857475.

[6] Yuanchang Liu and Richard Bucknall. A survey of formation control and motion planning of multiple unmanned vehicles. *Robotica*, 36(7):1019–1047, March 2018. ISSN: 1469-8668. DOI: 10.1017/s0263574718000218.

[7] Heng Su, Hongquan Yun, Jingwen He, Feng Zhang, and Yue Wang. Multi-aircraft path planning method based on cooperative search a-star algorithm. *2019 IEEE International Conference on Unmanned Systems (ICUS)*, 2019. DOI: 10.1109/icus48101.2019.8995994.

[8] Luitpold Babel. Coordinated target assignment and uav path planning with timing constraints. *Journal of Intelligent & Robotic Systems*, 94(3–4):857–869, July 2018. ISSN: 1573-0409. DOI: 10.1007/s10846-018-0910-9.

[9] Zhihong Liu, Xiangke Wang, Lincheng Shen, Shulong Zhao, Yirui Cong, Jie Li, Dong Yin, Shengde Jia, and Xiaojia Xiang. Mission-oriented miniature fixed-wing uav swarms: a multilayered and distributed architecture. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(3):1588–1602, March 2022. ISSN: 2168-2232. DOI: 10.1109/tsmc.2020.3033935.

[10] Yeongho Song, Seunghan Lim, Hyunsam Myung, Hokeun Lee, Junho Jeong, Heungsik Lim, and Hyondong Oh. Distributed swarm system with hybrid-flocking control for small fixed-wing uavs: algorithms and flight experiments. *Expert Systems with Applications*, 229:120457, November 2023. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2023.120457.

[11] Varun Madabushi, Yocheved Kopel, Adam Polevoy, and Joseph Moore. Dense fixed-wing swarming using receding-horizon nmpc. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8656–8662. IEEE, May 2025. DOI: 10.1109/icra55743.2025.11127916.

[12] Madhavan Shanmugavel, Antonios Tsourdos, Rafal Zbikowski, and Brian White. Path planning of multiple uavs with clothoid curves in two dimensions. *IFAC Proceedings Volumes*, 40(7):461–466, 2007. ISSN: 1474-6670. DOI: 10.3182/20070625-5-fr-2916.00079.

[13] Wang Yu, Wang Shuo, Wang Rui, and Tan Min. Generation of temporal–spatial bezier curve for simultaneous arrival of multiple unmanned vehicles. *Informa-*

http://www.imavs.org/

tion Sciences, 418–419:34–45, December 2017. ISSN: 0020-0255. DOI: 10.1016/j.ins.2017.07.031.

[14] Landon Shumway and Randal W. Beard. Time-synchronized b-spline path planning for multi-agent uav systems with fixed speed profiles. In *2025 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 272–278. IEEE, May 2025. DOI: 10.1109/icuas65942.2025.11007834.

[15] Madhavan Shanmugavel, Antonios Tsourdos, Rafal Zbikowski, and Brian White. 3d path planning for multiple uavs using pythagorean hodograph curves. In *AIAA Guidance, Navigation and Control Conference and Exhibit*. American Institute of Aeronautics and Astronautics, June 2007. DOI: 10.2514/6.2007-6455.

[16] Venanzio Cichella, Ronald Choe, Bilal S. Mehdi, Enric Xargay, Naira Hovakimyan, Anna C. Trujillo, and Isaac Kaminer. Trajectory generation and collision avoidance for safe operation of cooperating uavs. In *AIAA Guidance, Navigation, and Control Conference*. American Institute of Aeronautics and Astronautics, January 2014. DOI: 10.2514/6.2014-0972.

[17] Ronald Choe, Javier Puig-Navarro, Venanzio Cichella, Enric Xargay, and Naira Hovakimyan. Cooperative trajectory generation using pythagorean hodograph bézier curves. *Journal of Guidance, Control, and Dynamics*, 39(8):1744–1763, August 2016. ISSN: 1533-3884. DOI: 10.2514/1.g001531.

[18] Lester E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497, July 1957. ISSN: 0002-9327. DOI: 10.2307/2372560. URL: http://www.jstor.org/stable/2372560 (visited on 07/02/2025).

[19] Madhavan Shanmugavel, Antonios Tsourdos, Rafał Żbikowski, and Brian White. Path planning of multiple uavs using dubins sets, 2005. DOI: 10.2514/6.2005-5827.

[20] Qing-yang Chen, Ya-fei Lu, Gao-wei Jia, Yue Li, Bing-jie Zhu, and Jun-can Lin. Path planning for uavs formation reconfiguration based on dubins trajectory. *Journal of Central South University*, 25(11):2664–2676, November 2018. ISSN: 2227-5223. DOI: 10.1007/s11771-018-3944-z.

[21] Aditya K. Rao, Kushal P. Singh, and Twinkle Tripathy. Curvature bounded trajectories of desired lengths for a dubins vehicle. *Automatica*, 167:111749, September 2024. ISSN: 0005-1098. DOI: 10.1016/j.automatica.2024.111749.

[22] Zheng Chen, Kun Wang, and Heng Shi. Elongation of curvature-bounded path. *Automatica*, 151:110936, May 2023. ISSN: 0005-1098. DOI: 10.1016/j.automatica.2023.110936.

[23] Weiran Yao, Naiming Qi, Jun Zhao, and Neng Wan. Bounded curvature path planning with expected length for dubins vehicle entering target manifold. *Robotics and Autonomous Systems*, 97:217–229, November 2017. ISSN: 0921-8890. DOI: 10.1016/j.robot.2017.09.003. URL: https://www.sciencedirect.com/science/article/pii/S092188901730132X.

[24] Richard Peirce Brent. *Algorithms for minimization without derivatives*. Prentice-Hall, 1973.

[25] E. R. Hansen. Global optimization using interval analysis: the one-dimensional case. *Journal of Optimization Theory and Applications*, 29(3):331–344, 1979. ISSN: 1573-2878. DOI: 10.1007/BF00933139.

[26] Q. Huangfu and J. A. J. Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, 2018. ISSN: 1867-2957. DOI: 10.1007/s12532-017-0130-5. URL: https://doi.org/10.1007/s12532-017-0130-5.

[27] Fanchen Wu and Zheng Chen. Minimum-time paths for dubins airplane in steady wind. *Journal of Guidance, Control, and Dynamics*:1–13, August 2025. ISSN: 1533-3884. DOI: 10.2514/1.g008932.

[28] Gautier Hattenberger, Murat Bronz, and Michel Gorraz. Using the Paparazzi UAV System for Scientific Research. In *IMAV 2014, International Micro Air Vehicle Conference and Competition 2014*, pp 247–252, Delft, Netherlands, August 2014. DOI: 10.4233/uuid:b38fbdb7-e6bd-440d-93be-f7dd1457be60. URL: https://enac.hal.science/hal-01059642.

[29] Weijia Yao, Héctor Garcia de Marina, Zhiyong Sun, and Ming Cao. Guiding vector fields for the distributed motion coordination of mobile robots. *IEEE Transactions on Robotics*, 39(2):1119–1135, April 2023. ISSN: 1941-0468. DOI: 10.1109/tro.2022.3224257.

[30] Mark Owen, Randal W. Beard, and Timothy W. McLain. *Implementing dubins airplane paths on fixed-wing uavs**. In *Handbook of Unmanned Aerial Vehicles*. Kimon P. Valavanis and George J. Vachtsevanos, editors. Springer Netherlands, Dordrecht, August 2014, pages 1677–1701. ISBN: 978-90-481-9707-1. DOI: 10.1007/978-90-481-9707-1_120.

http://www.imavs.org/

Center-of-Gravity Placement Effects on a Bio-Inspired Flapping-Wing MAV: An Empirical Characterization for Sensor Payload Integration

I. Guerra Bello[1], A. Lopez[2]*, and M. Gallo[1]
[1]Inholland University of Applied Sciences, Delft, the Netherlands
[2]TU Delft, MAVLab, Delft, the Netherlands

http://www.imavs.org/

Abstract

Autonomous aerial inspection of commercial greenhouses requires a micro air vehicle (MAV) capable of flying close to the canopy while carrying a sensing payload for pest detection. The Flapper Nimble+, a bio-inspired flapping-wing MAV, is well suited to this role, but integrating sensors onto the platform is non-trivial: every control moment is generated through the wings, so even a small shift in the centre of gravity (CG) changes the trim the controller must hold and the way the airframe oscillates. Without quantitative guidance on where payload mass may be placed, integration is reduced to trial and error. We report a systematic two-axis CG-placement campaign on the Nimble+ in which mass was swept along the fore–aft and vertical body axes across 23 configurations and 64 hovering flights, logged simultaneously from onboard flight-controller (FC) telemetry and an external motion-capture system. Two robust empirical relations emerge: steady-state pitch trim varies near-linearly with fore–aft CG offset at -0.97 deg/mm, and body-pitch oscillation amplitude rises near-linearly with the magnitude of vertical CG offset at $+0.60$ deg/mm. These relations give designers a placement rule: keep the payload CG close to the bare-platform CG fore–aft to minimise trim demand, and with minimal vertical offset from it to minimise oscillation. An integrated sensing build assembled for greenhouse inspection confirms the fore–aft relation: its measured pitch-trim shift matches the prediction to within the bare-platform flight-to-flight spread. The vertical relation is corroborated in direction but, owing to the larger variability of the oscillation amplitude, is less precisely predictive. A mount-position imaging experiment shows that placing the camera close to the CG improves detection rate by a factor of 1.78.

*Email: a.p.lopez@tudelft.nl

1 Introduction

Commercial greenhouse crops must be inspected for pests at the founding-colony stage, before an infestation spreads – yet this early detection is hard to achieve with current methods. Pest infestations in high-density plantings of tomato, cucumber and sweet bell pepper are most effectively controlled by biological agents deployed before a colony reaches damaging density, yet the dominant monitoring methods – periodic manual scouting and passive sticky traps – are too infrequent and too coarse to catch a founding colony reliably [1]. Ground robots have been developed for greenhouse tasks but are constrained to row-level access and cannot inspect elevated canopy zones or leaf undersides where many pests establish [2]. An aerial platform that can fly close to the canopy, cover a full corridor pass, and carry a camera capable of resolving pest clusters would close this gap.

The Flapper Nimble+, a bio-inspired flapping-wing MAV, is well suited to this role: its flapping produces lower downwash and a lower acoustic profile than rotary-wing alternatives, reducing disturbance to foliage during close-range operation; its light wings pose little risk of damaging crops or greenhouse structure on incidental contact. It has been demonstrated in a greenhouse environment carrying a camera for pest detection, though that work concluded that a lighter, better-integrated payload was needed [3]. Integrating sensors onto the platform is, however, non-trivial [4]. The Nimble+ generates every control moment through its wings alone [5], so the CG position appears as a parameter in every rotational equation of motion. Adding a sensor payload simultaneously increases total mass and shifts the CG, and the shift is the subtler effect: a few millimetres changes the trim the controller must hold and the amplitude at which the airframe oscillates [6, 7], directly affecting both flight endurance and the quality of images the payload can capture. While the control physics of the platform is established [5], no quantitative guidance exists for how much a given CG shift costs in trim or oscillation on the Nimble+, leaving sensor integration as trial and error.

This paper provides that guidance. We report a systematic two-axis CG-placement campaign and derive two empirical relations that form a bounded predictive model for any payload configuration within the tested envelope. We validate the

http://www.imavs.org/

model against a fully integrated sensing build for greenhouse inspection – quantitatively for the fore–aft trim relation and directionally for the vertical oscillation relation – and show, through a mount-position imaging experiment, that the same CG placement that minimises trim cost also maximises in-flight camera detection rate.

2 PLATFORM AND MOMENT GENERATION

The Flapper Nimble+ is a bio-inspired flapping-wing MAV with a published payload capacity of 25 g above an all-up mass of roughly 102 g,a hover wing-beat frequency of approximately 14-17 Hz, a 5 V / 400 mA payload rail, and an onboard Crazyflie Bolt 1.1 flight controller running a PID stabilisation loop [8]. The platform descends from the DelFly lineage [9] and carries no tail surface: all three-axis control torque is produced through differential wing kinematics via four inputs [5]: left- and right-wing flapping frequencies f_L and f_R, symmetric dihedral angle Γ, and yaw wing-root deflection δ_ψ. The body axis convention is shown in Figure 1: Y along the wing leading edge (LE), Z along the spine, X out of the wing plane.

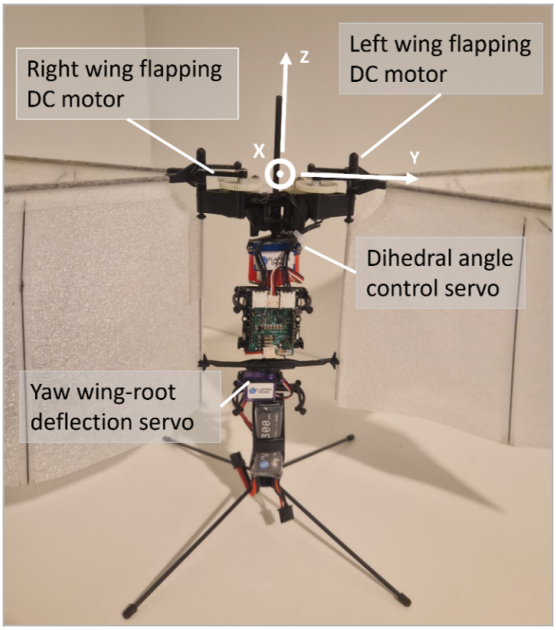


Figure 1: Body axis convention and control actuators of the Flapper Nimble+.

Because the platform is tailless, every control moment equals an aerodynamic force times its arm to the CG [5]. For pitch, the dihedral servo tilts the flapping plane about a hinge ahead of the CG, displacing the thrust vector by $r_x \sin\Gamma$, giving

$$\tau_\Gamma = T\, r_x \sin\Gamma, \tag{1}$$

where T is hover thrust and r_x the fore–aft arm from CG to hinge. A forward CG shift reduces r_x, weakening pitch authority and requiring a larger Γ to hold trim [10] — the physical origin of the sensitivity characterised in Section 4. Roll uses the lateral wing offsets; yaw uses asymmetric wing-root twist. In every case the control moment depends on the CG-to-force arm, so any payload shift changes that arm and propagates into trim, oscillation, and ultimately the quality of the captured imagery.

3 CG SWEEP METHODOLOGY

3.1 Test environment and flight protocol

All flights were conducted in the CyberZoo, a $10\times10\times7$ m enclosed flight volume at TU Delft equipped with an OptiTrack motion-capture (MoCap) system at up to 180 Hz. Every flight was logged by two independent streams: the MoCap returning aircraft position and attitude from infrared markers, and the onboard FC telemetry logging attitude estimates, gyro rates, motor commands and battery state at 50 Hz.

The FC runs a PID stabilisation loop using a downward time-of-flight (ToF) sensor for altitude hold and the onboard IMU for attitude. Pitch and yaw received no pilot input throughout the campaign, so both channels reflect the controller's unforced response to each CG condition. A free-hover protocol was used: the aircraft was launched from one corner and allowed to drift forward under its natural pitching bias, with the pilot giving only occasional lateral corrections to stay within the volume.

Each configuration was flown three times on a single fully charged battery, holding state-of-charge effects constant within a configuration. The campaign spanned several days; on each day the unmodified bare-platform (BASE) configuration was re-flown first and used as that session's reference, ensuring all comparisons are within-session.

3.2 CG mounts and tested configurations

CG offsets were imposed using 3D-printed mass-mount brackets (Figure 2). Front and aft brackets bolted to the FC imposed fore–aft (X) offsets; a top disc and a bottom washer stack imposed vertical (Z) offsets. Each configuration's CG was computed analytically from bench-weighed masses and CAD-extracted mount centroids. Pure single-axis displacement is geometrically impossible, so each configuration carries a small residual in the orthogonal axis; these are used explicitly in the cross-axis checks of Section 4.

The X-sweep covered 14 configurations spanning $\Delta X_{CG} \in [-1.75, +2.86]$ mm (0–11.8 g added mass). The Z-sweep covered 9 configurations spanning $\Delta Z_{CG} \in [-17.55, +10.87]$ mm (0–19.8 g); the vertical range is intentionally exaggerated to map the limits of the relation. Both sweeps are summarised in Table 1.

3.3 Per-flight metrics

The two data streams originate from independent clocks and are processed separately. For each flight a hover window is extracted by isolating the in-flight segment and trim-

http://www.imavs.org/



Figure 2: Mass-mount brackets: fore–aft sweep (front (right) and aft (left) brackets) and vertical sweep (top disc and bottom washer stack).

ming take-off and landing transients; flights with fewer than 15 s of retained data are excluded. From the FC telemetry three metrics are derived. The mean pitch trim $\overline{\theta}_p$ is the time-averaged body pitch, capturing the steady-state attitude the controller settles on. The pitch servo is driven by a pulse-width-modulation (PWM) command m_1. The pitch-servo total variation

$$\mathrm{TV}_{m1} = \frac{1}{T_{\mathrm{hov}}} \sum_{i=1}^{N-1} \left| m_1[i{+}1] - m_1[i] \right| \ [\mathrm{PWM\,s^{-1}}] \qquad (2)$$

Sweep	Configs	Δ_{CG} range (mm)	Mass (g)
X (fore–aft)	14	-1.75 to $+2.86$	0–11.8
Z (vertical)	9	-17.55 to $+10.87$	0–19.8

Table 1: CG-sweep ranges; each configuration flown three times.

measures cumulative pitch-servo effort per second. The servo reversal rate

$$\nu_{\mathrm{cross}} = \frac{1}{T_{\mathrm{hov}}} \sum_{i=1}^{N-1} \mathbf{1}\big[(m_1[i]{-}m_1^*)(m_1[i{+}1]{-}m_1^*) < 0\big] \ [\mathrm{s^{-1}}] \qquad (3)$$

counts zero-crossings of the servo-neutral reference m_1^* per second and is sensitive to the direction of any trim bias. From the MoCap quaternion, body pitch is derived and the oscillation amplitude σ_{θ_p} is computed as the standard deviation of the mean-removed pitch signal. The extended-Kalman-filter (EKF) smoothed FC signal is used for $\overline{\theta}_p$ and servo metrics; the unfiltered MoCap signal is used for σ_{θ_p}.

4 Results

4.1 *Fore–aft axis: pitch-trim demand*

Of the X-sweep configurations, 38 valid FC flights and 40 valid MoCap flights passed the hover-window filters. The primary finding is a near-linear relationship between per-flight mean body pitch and ΔX_{CG} (Figure 3). An ordinary least-squares (OLS) fit to the per-configuration means gives

$$\overline{\theta}_p(\Delta X_{CG}) = -1.86 - 0.97\,\Delta X_{CG} \quad [\mathrm{deg}], \qquad (4)$$

with Pearson correlation coefficient $r = -0.97$: approximately one degree of nose-up trim per millimetre of forward CG shift. The -1.86 deg intercept is the residual trim of this build at $\Delta X_{CG} = 0$.

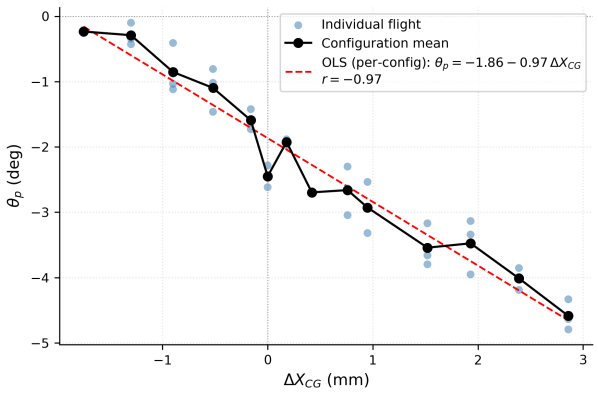


Figure 3: Per-flight mean body pitch vs. ΔX_{CG}. Dots are individual flights; connected points are per-configuration means; dashed line is the OLS fit of Equation 4.

The pitch-axis control activity provides independent corroboration (Figure 4). TV_{m1} rises away from $\Delta X_{CG} = 0$ in both directions, the expected signature of the servo doing more total work as $|\Delta X_{CG}|$ grows. ν_{cross} behaves asymmetrically: forward CG biases m_1 above neutral so zero-crossings become rare even as TV_{m1} peaks; at the aft extreme the trim approaches neutral and reversals are highest. This asymmetry is the control-side fingerprint of Equation 4,

read from the raw PWM stream without passing through the state estimator.

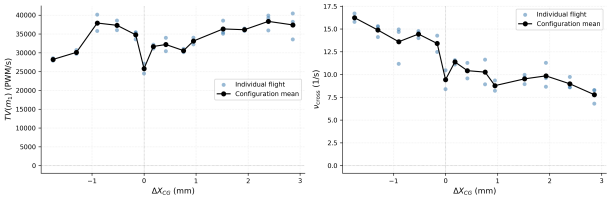


Figure 4: Pitch-servo total variation (left) and reversal rate (right) vs. ΔX_{CG}. Both independently corroborate the trim relation.

A cross-axis check confirms the relation is fore–aft specific. Mean pitch from the Z-sweep varies by only $\sim$1.4 deg across the full $\pm$17 mm vertical range, consistent with the small residual ΔX_{CG} carried by the Z mounts propagated through Equation 4. Vertical CG placement does not drive steady-state pitch trim.

4.2 *Vertical axis: pitch-oscillation amplitude*

Of the Z-sweep configurations, 26 valid FC flights and 21 valid MoCap flights passed the filters.

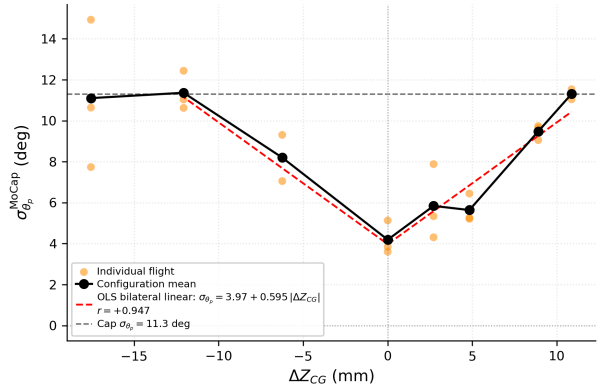


Figure 5: Pitch-oscillation amplitude vs. ΔZ_{CG}. Dashed line: bilateral OLS fit; horizontal dashed line: saturation cap.

Figure 5 shows a bilateral, near-monotonic rise in σ_{θ_p} with $|\Delta Z_{CG}|$. The per-configuration means rise from $\sigma_{\theta_p} \approx 4.2$ deg at $\Delta Z_{CG} = 0$ to about 11.3 deg at both vertical extremes, a near-threefold increase on both sides. The rise is bilateral: vertical offsets above and below the bare-platform CG both amplify the oscillation.

Because the platform is flown closed-loop, what is measured is the combined response of the airframe and its PID stabilisation loop rather than the airframe alone, and two effects plausibly contribute. The vertical offset alters the arms and inertia through which the wings generate the pitch moment, and the PID, tuned to the bare-platform response, applies the same commands to an airframe that now reacts differently and so operates away from its tuning point. Separating the two would require closed-loop system identification, which was outside this campaign. What the data shows unambiguously is that vertical CG displacement in either direction amplifies pitch oscillation. An OLS fit gives

$$\sigma_{\theta_p}^{\text{pred}}(\Delta Z_{CG}) = \min\big(3.97 + 0.595|\Delta Z_{CG}|,\ 11.3\big)\ [\text{deg}], \tag{5}$$

with $r = +0.95$ and a saturation cap at 11.3 deg. The single slope on $|\Delta Z_{CG}|$ is treated as an empirical engineering fit, justified by the symmetric, bilateral response noted above rather than by a single physical mechanism. The vertical brackets also add mass with $|\Delta Z_{CG}|$ (up to $\sim$19% of body mass), and the resulting parallel-axis inertia increase is small by comparison ($\sim$0.6–1.8% of $I_{yy,0} = 3.35 \times 10^{-4}$ kg m^2 [8] at the sweep extremes), so Equation 5 is best read as a fit to mass-and-offset-coupled configurations rather than a pure CG-height effect, which is a further reason the vertical relation is confirmed in direction but not tightly in magnitude.

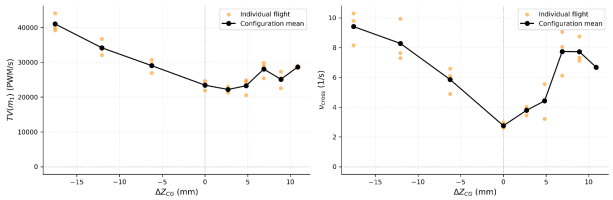


Figure 6: Pitch-servo total variation (left) and reversal rate (right) vs. ΔZ_{CG}.

Servo activity corroborates this: both TV_{m1} and ν_{cross} rise away from $\Delta Z_{CG} = 0$, steeper on the lower side. A cross-axis check shows σ_{θ_p} varying by only $\sim$3 deg across the fore–aft range, confirming that pitch-oscillation amplitude is vertically driven and fore–aft independent.

5 Trim Model

Equations 4 and 5 are packaged into a simple payload-placement model. The system CG is computed by mass-weighted combination of the bare-platform CG with the payload-component centroids; each component is treated as a point mass (its own area moment is one to two orders of magnitude below the parallel-axis contribution [5]), and the inertia tensor is updated about the new CG. The model returns predicted pitch trim and oscillation amplitude, valid within the tested envelope:

$$\begin{aligned} -1.75 \le \Delta X_{CG} \le +2.86\ \text{mm}, \\ -17.55 \le \Delta Z_{CG} \le +10.87\ \text{mm}. \end{aligned} \tag{6}$$

Configurations outside this envelope are flagged as unsupported extrapolations. A compact inspection payload sits well inside both bounds. Of the two outputs, the pitch-trim prediction is the quantitatively reliable one; the oscillation ampli-

http://www.imavs.org/

http://www.imavs.org/

tude should be read as an indicative magnitude and a dependable indicator of trend rather than a tight point prediction.

6 Integrated-Payload Comparison

To test the model against a real build, a sensing configuration was assembled: an OpenMV AE3 global-shutter camera (1 MP, 120 fps, 5 g, integrated 8×8 ToF sensor & IMU), a downward optical-flow/ToF sensor, and a ultra-wideband (UWB) positioning deck + ToF sensor pack, totalling 15.1 g. The build shifts the system CG by $\Delta X_{CG} = -1.32$ mm and $\Delta Z_{CG} = -0.69$ mm (1.50 mm total), which sits inside the envelope of Equation 6.

The integrated build and the bare platform were each flown six times under the same free-hover protocol. Table 2 compares the model-predicted change (integrated minus bare) against the measured within-session change. Because the bare-platform trim baseline drifts between sessions, only within-session deltas are compared; the relation is predictive on the deltas despite this baseline drift.

Quantity	Predicted Δ	Measured Δ	Residual
Pitch trim (deg)	+1.29	+1.11	−0.18
Pitch oscillation (deg)	+0.41	+0.83	+0.42

Table 2: Model-predicted versus measured within-session change (integrated minus bare platform), second test session, $n = 6$ bare and $n = 6$ integrated flights (oscillation row: $n = 5$ bare, one flight excluded for a motion-capture tracking fault).

Predicted and measured pitch trim agree in sign and magnitude: the model predicts a +1.29 deg nose-up shift for the −1.32 mm aft CG offset, against a measured +1.11 deg, a residual (−0.18 deg) within the bare-platform flight-to-flight spread ($\sigma = 0.19$ deg). The integrated build therefore does not measurably degrade static stability, and the trim relation of Equation 4 carries over from the controlled sweep to a real build.

The oscillation row of Table 2 is reported for completeness but is a weak test of Equation 5: the build's vertical CG shift is small ($\Delta Z_{CG} = -0.69$ mm), so the predicted change (+0.41 deg) is itself comparable to the flight-to-flight scatter. The measured amplitude rose by +0.83 deg, in the predicted direction but overshooting the prediction, which is consistent with the oscillation relation being established by the full ±17 mm sweep of Section 4 rather than by this near-centred build. This is the expected asymmetry between the two relations: fore–aft trim is predicted to within its flight-to-flight spread, whereas vertical oscillation amplitude is confirmed in direction but not closely in magnitude.

6.1 Camera mount position and detection rate

So far CG placement has been treated as a flight-performance question. It is also a sensor-performance question: where the camera sits relative to the CG sets how much of the airframe's pitch motion affects image acquisition via motion blur.

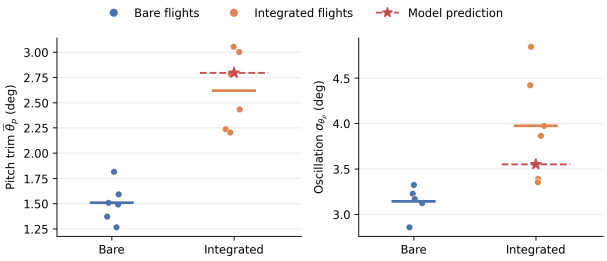
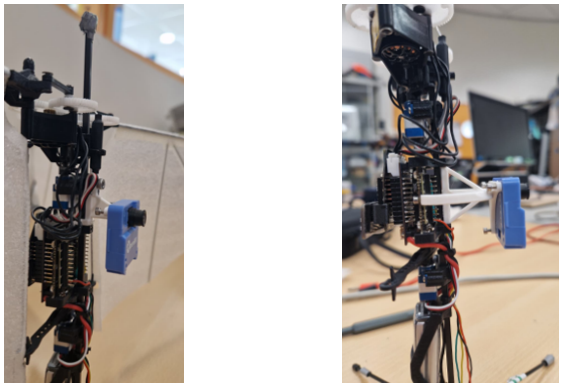


Figure 7: Integrated-build validation. For each quantity, bare-platform (blue) and integrated-build (orange) per-flight values are shown with their means (horizontal bars); the red star marks the model-predicted integrated value (bare mean plus predicted change). Trim matches the prediction within scatter; oscillation rises as predicted in direction but overshoots in magnitude.

To show this, the AE3 camera was flown over a fixed AprilTag target during hover in two mounting positions (Figure 8). In the first the camera sits close to the system CG; in the second it is mounted about 20 mm lower, at a longer arm from the CG.

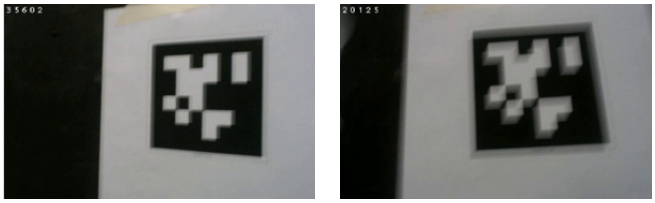


Near-CG mount Below-CG mount

Figure 8: Side views of the two camera mounting positions tested. The near-CG mount (left) places the camera close to the system CG; the below-CG mount (right) places it about 20 mm lower.

What sets the two mounts apart is the magnitude of the total CG shift from the bare-platform reference: 1.16 mm for the near-CG mount and 2.70 mm for the below-CG mount, about 2.3× larger. Tag detections were extracted offline over the in-window hover frames. The near-CG mount detected the tag in 24.3% of frames against 13.6% for the below-CG mount, a factor of 1.78.

At a larger arm l from the CG, a given body angular rate ω produces a proportionally larger linear velocity at the im-



Near-CG mount Below-CG mount

Figure 9: Flying image results from the two mounts. Near the CG (left), below the CG (right)

age plane ($v \approx \omega l$), so the below-CG camera sweeps more image motion into each frame and blurs a larger fraction of them. In short, the larger the body angular rate the camera experiences, the more it degrades the captured image; mounting the camera close to the CG therefore minimises this image-plane motion and maximises detection rate, the same placement that minimises trim demand. Figure 9 shows this directly – the same tag imaged sharply from the near-CG mount and smeared from the below-CG mount.

7 CONCLUSION

We characterised how centre-of-gravity placement affects flight on a bio-inspired flapping-wing MAV through a controlled two-axis mass-placement sweep on the Flapper Nimble+. Fore–aft CG offset drives steady-state pitch trim near-linearly at -0.97 deg/mm ($r = -0.97$), and vertical CG offset drives pitch-oscillation amplitude at $+0.60$ deg/mm ($r = +0.95$); each relation is corroborated by independent servo-activity metrics and confirmed axis-specific by cross-axis checks. The reported slopes and intercept are calibration constants of this specific airframe and should not be transferred numerically to other platforms; what is expected to generalise is the underlying mechanism – CG-referenced pitch trim scaling and vertical-offset-driven oscillation (Equation 1) – rather than its magnitude, consistent with qualitatively similar CG sensitivities reported for other tailless FW-MAVs [6, 7]. Packaged as a bounded linear model, the fore–aft relation predicts the pitch trim of an integrated greenhouse-inspection build to within the bare-platform flight-to-flight spread, validating it as a quantitative placement tool. The vertical relation is confirmed in direction–oscillation amplitude rises with vertical CG offset in both the sweep and the integrated build–but its larger flight-to-flight variability makes the precise amplitude harder to predict, so it is best used as a bounded trend. The practical rule is direct: place payload mass close to the bare-platform CG fore–aft to minimise trim demand, and with minimal vertical offset from it to minimise oscillation. Future work centres on tightening the oscillation prediction through an expanded integrated-build campaign under a stabilised position hold and closed-loop system identification, alongside a payload-aware feedforward trim schedule and autonomous corridor flight with the integrated build inside a commercial greenhouse.

ACKNOWLEDGEMENTS

This publication is part of the project 'Bio-inspired Micro Aerial Vehicle for sustainable pest control in horticulture – Flappy', with file number RAAK.PRO04.081, of the research program RAAK-PRO, which is financed by Regieorgaan SIA, part of the Netherlands Organisation for Scientific Research (NWO).

REFERENCES

[1] J.C. van Lenteren. The state of commercial augmentative biological control: plenty of natural enemies, but a frustrating lack of uptake. *BioControl*, 57(1):1–20, 2012.

[2] C.W. Bac, E.J. van Henten, J. Hemming, and Y. Edan. Harvesting robots for high-value crops: state-of-the-art review and challenges ahead. *Journal of Field Robotics*, 31(6):888–911, 2014.

[3] O. Awesta, V. Hollink, and M. Gallo. Flapping-wing drones for pests and diseases detection in horticulture. *IEEE Transactions on AgriFood Electronics*, pages 1–10, 2025.

[4] M.L. Afakh, A.A. Saputra, H. Sato, K. Wada, and N. Takesue. Servo-driven flapping-wing aerial vehicle (FWAV): Payload capacity and navigation performance. In *2025 IEEE/SICE International Symposium on System Integration (SII)*, pages 1071–1076, 2025.

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

[6] L.T.K. Au and H.C. Park. Influence of center of gravity location on flight dynamic stability in a hovering tailless FW-MAV: Longitudinal motion. *Journal of Bionic Engineering*, 16(1):130–144, 2019.

[7] L.T.K. Au and H.C. Park. Influence of center of gravity location on flight dynamic stability in a hovering tailless FW-MAV: Lateral motion. *Journal of Bionic Engineering*, 17:148–160, 2020.

[8] Flapper Drones. Nimble+ specifications and documentation, 2026. `https://docs.flapper-drones.eu`.

[9] G.C.H.E. De Croon, M.A. Groen, C. De Wagter, B. Remes, R. Ruijsink, and B.W. Van Oudheusden. Design, aerodynamics and autonomy of the DelFly. *Bioinspiration & Biomimetics*, 7(2):025003, 2012.

[10] E. Sanchez-Laulhe, G.C.H.E. de Croon, and A. Ollero. Equilibrium state for a tailless flapping wing micro air vehicle in forward flight. *IEEE Robotics and Automation Letters*, 11(1):17–24, 2026.

http://www.imavs.org/

Pitch stabilisation and longitudinal control of a hummingbird-inspired drone using an active tail

Arno Melis*, Jeroen Vanpoucke, Thomas Roelandt, Jan Swevers, Dirk Vandepitte
KU Leuven, Celestijnenlaan 300 box 2420, 3001 Leuven, Belgium

http://www.imavs.org/

ABSTRACT

This paper examines the use of an active tail for pitch stabilisation and longitudinal control of a hummingbird-inspired flapping-wing drone. The tail is positioned underneath the wings such that actuation of the tail angle redirects the airflow underneath the wings, resulting in a pitch control torque. A tail model derived from actuator disc theory and the blade element method is integrated into a free-flight simulation framework for longitudinal flapping-wing flight. Simulation results demonstrate that a linear quadratic regulator effectively stabilises the system and tracks a desired pitch angle. By incorporating velocity and position PI controllers in a cascaded architecture, the drone is additionally able to follow a longitudinal position reference. Experimental tests on an unstable, pitch-only gimbal setup demonstrate that the linear quadratic regulator can successfully stabilise the system. The experimental results follows the same trend as the predictions from simulation, although additional oscillations are observed that do not appear in the simulated results.

1 INTRODUCTION

Engineers have drawn inspiration from the remarkable ability of insects and hummingbirds to recover from collisions and execute agile manoeuvres, leading to flapping-wing micro aerial vehicles (FWMAV). These drones mimic the characteristic front-to-back wing motion seen in their biological counterparts. Beyond agility, flapping-wing drones also offer improved safety compared to conventional rotary-wing aircraft. Their lightweight wings operate at lower frequencies than rotor blades resulting in less harmful collisions for both the drone and its surroundings.

Recent research on flapping-wing drones has yielded functional prototypes across a wide range of scales. At the larger end of the spectrum, the Flapper Nimble+ is a commercially available platform with a wingspan of 49 cm [1]. At the opposite extreme, the Robobee, with a wingspan of only 3.5 cm, is situated at the insect scale [2]. Between these scales, a number of hummingbird-sized drones with wingspans ranging from 15 to 25 cm have been developed [3, 4, 5]. A key challenge with insect-like flapping-wing flight is its inherent instability. Both in the longitudinal and lateral planes unstable oscillatory modes are present [6]. Active control is essential to stabilise the drone during flight. Nature offers several strategies, for example, insects and hummingbirds can shift their centre of gravity and modulate their wing motion to generate control torques. This paper investigates another mechanism used by hummingbirds: a tail [7]. Positioned underneath the wings, the tail can be angled to redirect airflow, which in turn produces a control torque. In addition, aerodynamic drag acting on the tail generates damping forces that can contribute to passive flight stability.

Previous studies have demonstrated that incorporating a tail into an FWMAV, either as an active control surface or as a passive stabilising element, can improve flight stability and facilitate trajectory tracking [8, 9, 10, 11, 12]. Existing approaches typically rely on largely empirical descriptions of tail-wake interactions combined with conventional PID-based controllers. As a result, the relative effectiveness of different modelling assumptions and control strategies has not yet been systematically established.

This work further explores the use of an active tail on a hummingbird-scale flapping-wing drone. The analysis focuses on the KUlibrie platform, developed at KU Leuven, which has a wingspan of 195 mm and a mass of 15.8 g [13]. This paper establishes an aerodynamic tail model for an FWMAV and integrates it into a free flight simulation of the KUlibrie. This simulation is used to design and validate a tail movement strategy capable of stabilising the drone pitch angle and controlling its position. A linear quadratic regulator (LQR) approach is applied for pitch stabilisation, which is experimentally validated on a test setup that only allows the drone to move around its pitching axis.

2 KULIBRIE DRONE PLATFORM

Figure 1 shows a render of the current prototype of the KUlibrie equipped with a tail. The wings consist of a Mylar foil attached to a carbon fibre frame, and are driven by a resonant actuation system [14]. Two DC motors independently drive the wings through a gear transmission. At the base of each wing, a torsion spring recovers kinetic energy by storing it as potential energy during stroke reversal and releasing it to support the subsequent stroke. To improve efficiency, the motors operate close to the system natural frequency, resulting in resonant wing motion. In the case of the KUlibrie, this resonant flapping frequency is 20 Hz.

*Email address: arno.melis@kuleuven.be

http://www.imavs.org/

The tail of the KUlibrie is actuated directly by a servo MASRTER DS 208 motor, which only allows tail rotation around the motor axis, parallel with the pitch axis. The tail surface is constructed from ROHACELL 51, a lightweight foam. The tail chord length varies along its span, with the largest chord located near the tips. This region underneath the wing tips experiences the strongest downward airflow generated by the flapping wings [13], resulting in higher aerodynamic force production when the tail surface is located there. The tail is mounted at 2.5 wing chord lengths behind the leading edge of the wings. The tail adds a mass of m_t = 5.2 g to the KUlibrie weight (1.7 g due to the motor, 1.1 g due to the tail surface, 2.4 g due to connection parts).
Wing and tail actuation are controlled by a SEEED XIAO nRF52840 Sense microcontroller, which incorporates a 3-axis gyroscope and a 3-axis accelerometer to estimate the drone attitude.

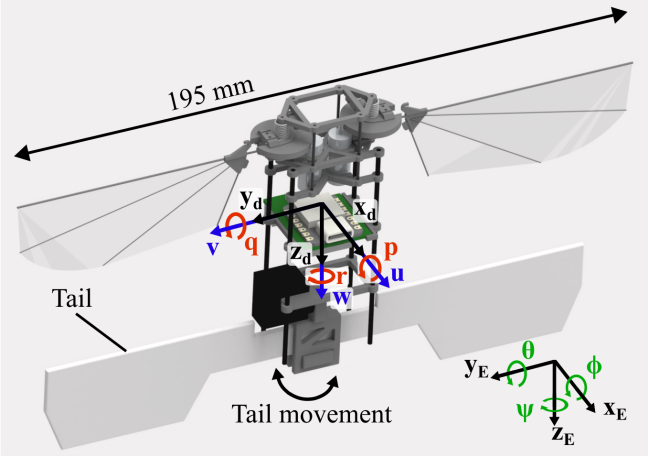


Figure 1: Render of the KUlibrie prototype with tail, with indication of body (x_d, y_d, z_d) and earth (x_E, y_E, z_E) coordinate system together with the velocity (u, v, w), angular velocity (p, q, r) and attitude (ϕ, θ, ψ) components

3 Flight Model

The flight model of the KUlibrie with tail describes the drone motion as the combined result of wing and tail actuation. It is based on the Newton's equations of motion, which determine how the drone velocity and position evolve under the influence of external forces. Following standard aircraft conventions, all kinematic and dynamic quantities are expressed in a body-fixed coordinate system attached to the drone. Figure 1 illustrates this coordinate frame, denoted (x_d, y_d, z_d), where the x_d-axis points forward along the body, the y_d-axis points to the right wing, and the z_d-axis points downward. Drone velocity components (u, v, w), angular velocity components (p, q, r), force components (X, Y, Z), and moment components (L, M, N) are all defined in this reference frame.

The longitudinal equations of motion in this body-fixed frame are given by:

$$\dot{u} = -qw + \frac{X_{\text{wings}}}{m} + \frac{X_{tail}}{m} - g\sin\theta, \tag{1}$$

$$\dot{w} = qu + \frac{Z_{\text{wings}}}{m} + \frac{Z_{tail}}{m} + g\cos\theta, \tag{2}$$

$$\dot{q} = \frac{M_{\text{wings}}}{I_y} + \frac{M_{tail}}{I_y}, \tag{3}$$

$$\dot{\theta} = q. \tag{4}$$

Here, I_y is the moment of inertia around the pitching axis (y_d), m the drone mass and g the gravitational acceleration. The forces and moments are decomposed into contributions from the wings and the tail.
These differential equations are implemented in simulation using a forward Euler integration scheme with a 1 ms time step. At each simulation time step, the aerodynamic models for the wings and tail compute their respective forces and moments, which are applied in the equations of motion to update the drone's velocities and orientation.

3.1 *Wing Forces*

A model based on the work of Wang et al. [15] determines the forces on the wings. The model is quasi-steady, meaning that wing forces only depend on the current velocity of the wing. It is also a blade element model, meaning that the wing is divided into thin chordwise sections and aerodynamic force contribution is calculated at each section. Figure 2 qualitatively illustrates such a chordwise element. In this study, the blade elements have a 0.1 mm width, and the aerodynamic force is assumed to be perpendicular to the wing surface. Previous research using the KUlibrie has already validated this model both using fluid simulations and experimental measurements [13, 16].

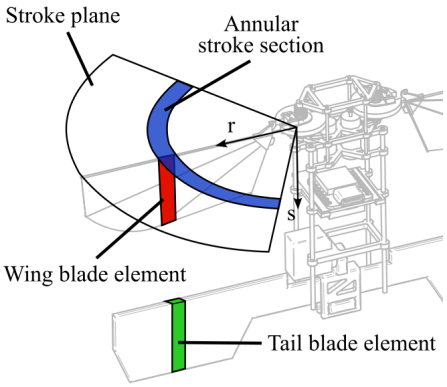


Figure 2: Illustration of a blade element of the wing and tail, together with the stroke plane sections

http://www.imavs.org/

3.2 Tail Forces

The forces on the tail are calculated in two steps, first the local downwards velocity of the air at the tail is estimated, and second, the force on the tail is estimated based on this local air velocity.

The velocity field of the air underneath the wings is estimated using actuator disc theory. This method uses the upward force on the wings to estimate the downward velocity imparted on the air, using momentum theory. Previous works used the entire stroke plane (cf. Figure 2) as the actuator disc [17], yielding the following expression for the downwards velocity component of the air underneath the wing:

$$V_z = \sqrt{\frac{T}{2\rho S}}. \tag{5}$$

With V_z the downward velocity component of the air, T the upward thrust generated by the wings, ρ the air density and S the total area of the stroke plane.

For a more detailed estimate of the downward velocity, the approach in this paper splits up the stroke plane up into annular contributions (cf. Figure 2). The upward force generated within each annular section is calculated using the blade element model of the wing forces. The local downward velocity component is:

$$V_z(r) = \sqrt{\frac{dT}{2\rho dS}}. \tag{6}$$

With dT the thrust generated by the annular wing section encompassed by the area dS, and located a distance r from the rotation point of the wing.

The previous formulations give the vertical air velocity at the wing leading edge. To estimate the velocity lower down at the tail position, actuator disc theory predicts that the velocity component rises until it has doubled its value at the leading edge. However, experimental measurements of the downward flow beneath flapping wings indicate that this evolution is no longer accurate beyond approximately two wing chord lengths below the wings [13, 18]. Moving further downwards, the downward velocity component becomes smaller. To implement this velocity profile in the model, a scaling factor of the maximum velocity is added based on experimental measurements on the KUlibrie. This gives the following model:

$$V_z(r, s) = \Omega(s)\sqrt{\frac{dT}{2\rho dS}}. \tag{7}$$

With s the vertical distance to the leading edge of the wing (cf. Figure 2), and Ω the scaling factor established from the measurements conducted in [13]. For the current tail location: $\Omega(2.5\overline{c}) = 1.4$.

A blade element model of the tail divides the tail into chordwise elements to estimate the tail force. The force contribution on each element is calculated using a flat plate model where the lift and drag coefficients are given by:

$$C_L = 2\sin(AoA)\cos(AoA), \tag{8}$$

$$C_D = 2\sin^2(AoA). \tag{9}$$

With AoA the local angle of attack of the tail blade element, resulting from the tail angle, local air velocity and tail velocity.

3.3 Actuation Dynamics

The actuation limits of the tail motor and wing motors are implemented in the simulation. First, the maximum rotation speed of the tail motor of 10 rad/s is prescribed in the simulation. For the wing movement, the simulation does not explicitly account for motor dynamics. Instead, wing motion is assumed to follow a roll-off square wave, as proposed in [19]. To avoid sudden jumps in wing velocity, a change in the stroke amplitude is only applied at the end of each up- or downstroke when the wing is instantaneously stationary. Finally, because the current tail design exceeds the current drone lift capacity, the tail mass in simulation is artificially reduced to the tail motor mass m_t = 1.7 g.

4 CONTROL METHODOLOGY

Figure 3 illustrates the control architecture used in this paper. The drone employs two control actions: adjustment of the tail angle δ_t generates a pitching torque, and modification of the wing stroke amplitude ϕ_m increases thrust. Together, these inputs control two longitudinal degrees of freedom. In this work, the two longitudinal position coordinates (x_E, z_E) are controlled trough a cascaded control structure. The position reference is first processed by the position controller, which outputs a desired velocity $(V_{x,\text{ref}}, V_{z,\text{ref}})$. This velocity reference is transformed into the drone body frame to obtain reference commands for u and w. Each value is sent to its respective controller: the w-controller determines the required stroke amplitude ϕ_m, while the u-controller computes a pitch reference θ_{ref}. This pitch reference is passed to the pitch controller, which calculates the necessary tail deflection δ_t.

4.1 Control Model

The full equations of motion from equations 1 - 4 are useful for simulation purposes, however they are too complex for direct use in controller design. A stroke-averaged linearised formulation makes the system tractable for controller design. This is obtained as follows: the forces are averaged over one stroke period and linearised around hover using their first order Taylor expansion. The movement in the z_E direction can be decoupled, leading to the following state space models [6]:

$$\begin{bmatrix} \dot{u} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{X_u}{m} & \frac{X_q}{m} & -g \\ \frac{M_u}{I_y} & \frac{M_q}{I_y} & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{X_\delta}{m} \\ \frac{M_\delta}{I_{yy}} \\ 0 \end{bmatrix} \delta_t, \tag{10}$$

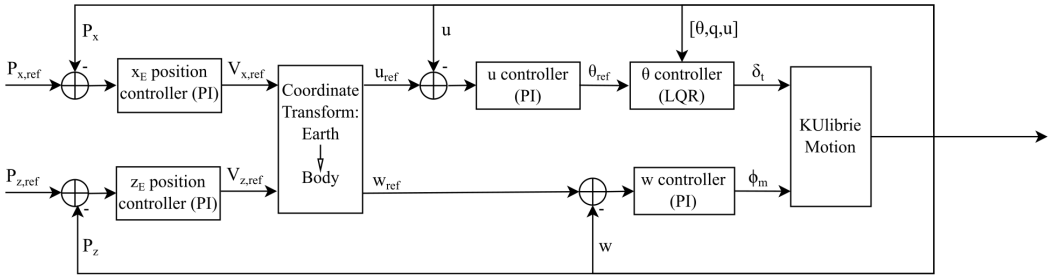


Figure 3: Block diagram of the cascaded controller structure

$$\dot{w} = \frac{Z_w}{m}\, w + \frac{Z_\phi}{m}\, \phi_m. \tag{11}$$

The parameters of this model such as Z_w represent a partial derivative of a stroke-averaged force component (here Z) with respect to a motion component or control input (here w). These quantities, commonly referred to as stability derivatives, each have a physical interpretation. Diagonal elements of the system matrix correspond to damping effects, whereas off-diagonal elements represent coupling between different directions of motion. The model parameters M_δ, X_δ and Z_ϕ describe the influence of the corresponding control actions on the system motion.
Because of the decoupling of w, the control problem can be separated such that the tail angle is used to control θ, while w is controlled through the stroke amplitude. This decoupling justifies the earlier division of the architecture into a w-controller and a θ-controller.

4.2 Pitch Controller Design

The longitudinal dynamics of the drone described by Equations 1 - 4 are unstable, exhibiting an unstable oscillatory mode in both u, θ and q [6]. To counteract this coupled oscillation, a state-feedback controller is employed, more specifically a linear quadratic regulator (LQR). Figure 4a. presents the block diagram of the LQR controller, which uses the system states θ, q, and u as feedback. The feedback matrix K is obtained by minimising the following cost function:

$$J = \int_0^\infty \left(\rho\, \theta^2(t) + \delta_t^2(t)\right)\, dt. \tag{12}$$

ρ is a weighting constant that balances tracking performance and control effort. An extension of the LQR controller is the linear quadratic integral (LQI) controller. Figure 4b. shows the block diagram of the LQI controller. This controller introduces an integral state to reduce steady-state errors. The inclusion of this integral state modifies the optimisation problem as follows:

$$J = \int_0^\infty \rho \left(\theta^2(t) + \alpha \left(\int_0^t e(\tau)\, d\tau\right)^2\right) + \delta_t^2(t)\, dt. \tag{13}$$

The additional weighting factor α determines the trade-off between the instantaneous error penalty and the accumulated error.

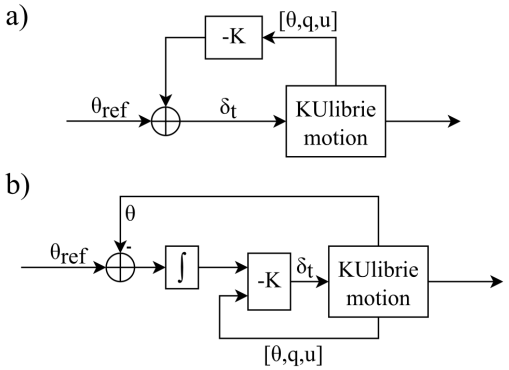


Figure 4: Block diagram of a) an LQR controller and b) an LQI controller

With both control types the state data is first passed through a moving average filter with a period of one stroke cycle before being fed back to the controller. This reduces the effect noise due to body oscillations of the drone. These rapid oscillations at the flapping frequency of the drone were averaged out during controller design, meaning that they represent unwanted noise on the feedback data.

4.3 Velocity and Position Controller Design

With the drone movement stabilised by the pitch controller, the remaining controllers in the architecture, namely the velocity and position controllers, act on systems that are already stable. For these subsystems, standard PI control is selected. The PI gains are tuned manually by balancing reference tracking accuracy against actuator effort, aiming at preventing actuator saturation for the given reference signals. Although the balance between speed and actuator limits depends on the specific application, in this paper the controller gains are adjusted to track a step input on the position to (0.5 m, -0.5 m) as closely as possible.

http://www.imavs.org/

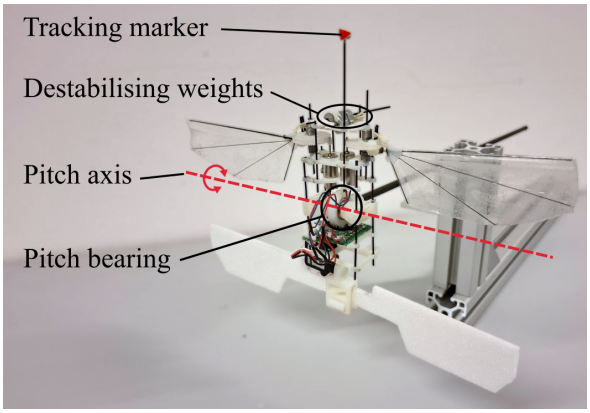


Figure 5: Experimental setup that only allows pitching rotation

4.4 Experimental Pitch Control Setup

Figure 5 shows the experimental setup to validate the LQR approach for pitch control. The KUlibrie is mounted on a ball bearing, restricting all degrees of freedom except the pitch rotation. Added weight on top of the drone moves its centre of gravity 2.1 mm above the fulcrum leading to an unstable system akin to free flight. This setup avoids crashes and isolates the performance of the pitch controller. However, because it only allows pitching movement, it does not fully capture the front-to-back unstable oscillating mode of free flight.

During the experiment θ and q are estimated using the onboard gyroscope and accelerometer together with a Kalman filter, the design of which is discussed in [20]. In addition, drone pitch angle is measured by visual tracking. This data is used after the experiment as an absolute measure for the estimation and tracking performance.

5 Results

5.1 Simulation Results

Figure 6 shows the simulation results of a step reference input of $\theta_{\text{reference}} = 3°$ applied at $t = 6$ s to the LQR pitch controller in free flight for three distinct values of the weighting factor $\rho = \{0.1,1,10\}$. It shows the pitch angle θ and tail angle δ_t of the drone during the free flight simulation. To ease interpretation, in this and following figures, the stroke averaged value of θ is shown. This removes the body oscillations, with an amplitude of about 3°. The results show that for all the displayed values of ρ the LQR controller is able to stabilise the pitch angle of the drone after a start-up transient. As expected, the higher values of ρ track the step input faster, but need larger tail movements. To this end, $\rho = 100$ and higher (not shown in the figure) would not stabilise the start-up transient as the tail actuator saturates. Each of the LQR controllers has a steady-state error due to the differences between the simulation and stroke-averaged linearised system.

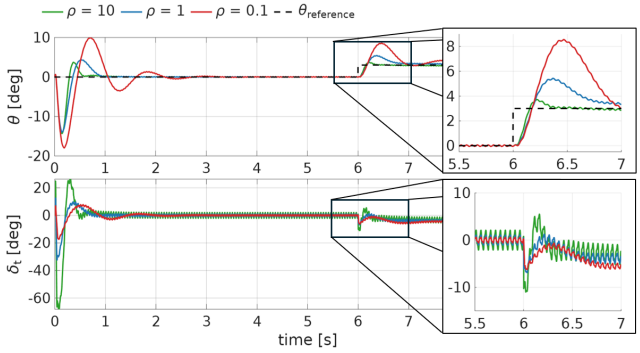


Figure 6: Simulation results of the LQR controller for $\rho = \{0.1, 1, 10\}$. The inserts show a zoom for the time interval [5.5 s; 7.5 s]

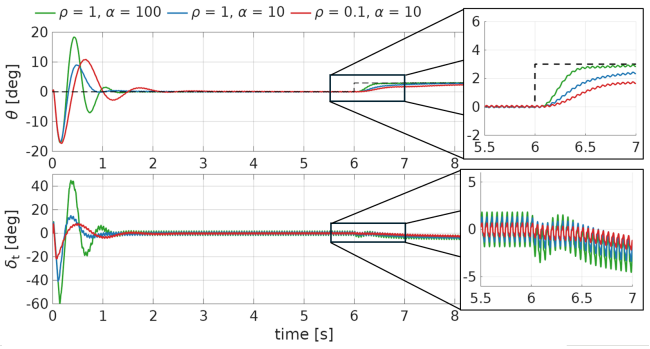


Figure 7: Simulation results of the LQI controller for $\rho = \{0.1, 1\}$ and $\alpha = \{10, 100\}$. The inserts show a zoom for the time interval [5.5 s; 7.5 s]

The results indicate that the steady-state error decreases as the value of ρ increases.

Figure 7 shows the simulation results of the same step reference applied to the LQI controller with various values of ρ and α. After a start-up transient, the LQI controller can also stabilise the free flight system. Again, higher values of ρ and α lead to better tracking performance, but larger tail movements. As opposed to the LQR controller, the LQI controller does not have a steady-state error, due to the integrator.

Figure 8 shows the simulation results of the cascaded control structure tracking a step position input of (0.5 m, -0.5 m), showing that the cascaded structure successfully tracks the reference. The results show that the settling time in the x_E-direction is 5.8 s with an overshoot of 15%. The tracking in the z_E-direction is faster, with a settling time of 3.6 s and no overshoot. Due to the integral action of the PI controller both axis show zero steady state error.

5.2 Experimental Results

Figure 9 and 10 show the experimental results of a ramp input reference of the LQR and LQI controller on the gimbal setup respectively. The ramp input reference goes from 0° to

http://www.imavs.org/

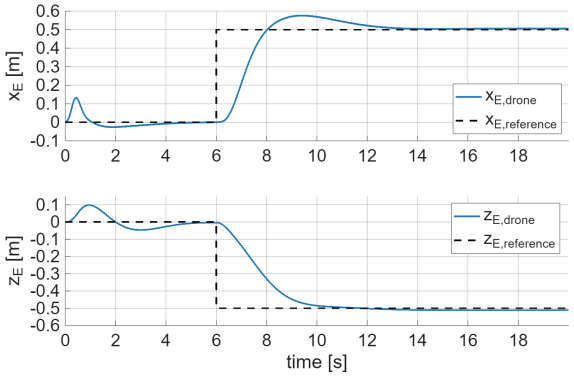


Figure 8: Simulation results of the cascaded controller tracking a (0.5 m, -0.5 m) step input

20° in 2 s. The figures show the estimated pitch angle through the Kalman filter, and the actual pitch angle measured through a camera equipped with visual tracking software. The results show that the Kalman filter estimates the pitch angle with good accuracy. Both the LQR and LQI controllers stabilise the system and track the reference. Despite this, added oscillations are observed, with a maximum amplitude of approximately 10°. The oscillation frequency is significantly lower than the flapping frequency of 20 Hz, suggesting that the pulsating wing forces are not the primary cause. A likely source of these oscillations are modelling errors. For instance, if the rotational damping in the model is underestimated, the resulting controller may be overly aggressive, leading to oscillatory behaviour. Other modelling inaccuracies, such as unmodelled tail motor dynamics or errors in the tail force model, may also contribute.

Comparing controller parameters, larger values of ρ and α result in a faster response, as expected. However, for $\rho = 10$ and $\alpha = 10$, the increased controller aggressiveness leads to larger oscillations. Conversely, lower values of ρ or α yield a slower response. In particular, the performance of the LQR controller with $\rho = 0.1$ is significantly degraded due to the system's slow reaction time.

Figure 11 shows the comparison of the experimental results, the prediction from the linear system and from the simulation. It shows that the linear system and the simulation are almost identical, differing by the simulation having a start-up transient. The experimental result shows roughly the same evolution as the simulation, however it has the added oscillations on top.

6 Study Limitations

The wing force model employed in this work is based on a quasi-steady blade element formulation, which assumes rigid wings and neglects three-dimensional aerodynamic effects. Although this modelling approach has been validated

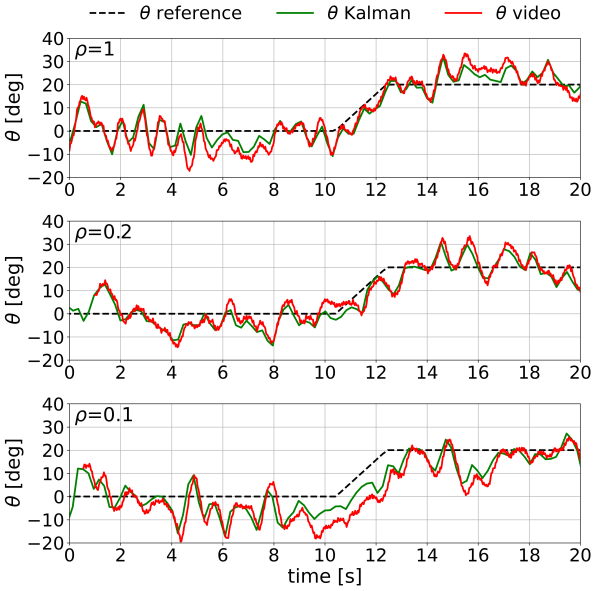


Figure 9: Experimental results of the LQI controller to a ramp input for values of $\rho = \{1, \frac{1}{5}, \frac{1}{10}\}$

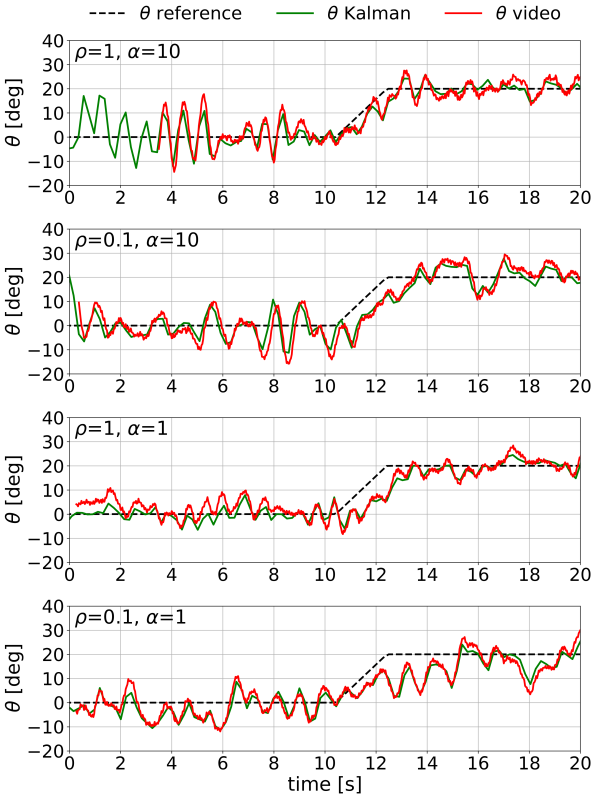


Figure 10: Experimental results of the LQI controller to a ramp input for $\rho = \{0.1, 1\}$ and $\alpha = \{1, 10\}$

http://www.imavs.org/

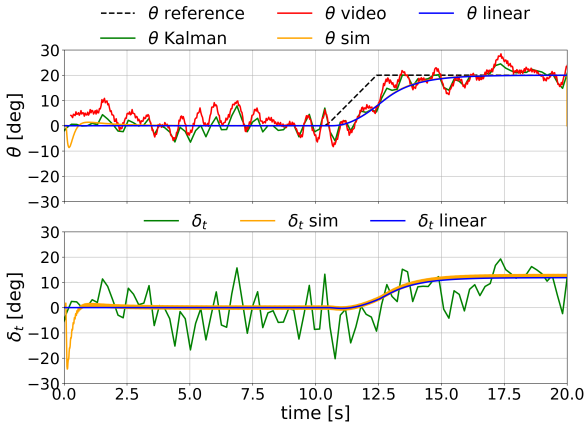


Figure 11: Comparison of the experimental, linear system and simulation results of the LQI controller

in both fluid simulations and static experiments [13, 16], the influence of forward flight velocity and wing deformation remains uncertain. The tail force model is founded on similar assumptions and additionally relies on actuator disc theory, which neglects vortex structures in the wake of the drone. Further detailed validation of this model is therefore required. Finally, while the pitch control strategy is experimentally validated in this study, the complete cascaded control architecture has yet to be validated experimentally.

7 Conclusion and Future Work

This paper analyses the longitudinal control of a hummingbird-inspired flapping-wing drone equipped with an active tail. The proposed tail is mounted beneath the wings and is directly actuated by a servo motor. A cascaded control architecture is explored, in which the position and velocity loops are implemented using PI controllers, while pitch control is achieved using a linear quadratic regulator. Controller gains are derived from a stroke-averaged, linearised free-flight model. The resulting controllers are evaluated in simulation using an actuator disc approach combined with a quasy-steady blade element method. Simulation results show that the LQR controller stabilises the system and tracks a step reference in pitch. However, differences between the stroke-averaged linear model and the higher-fidelity simulation introduce a steady-state error in pitch tracking. Augmentation of the LQR controller with integral action removes this steady-state error. Simulations of the complete cascaded control loop demonstrate that the system is capable of tracking a position reference of (0.5 m, -0.5 m).
Experimental validation of the pitch control strategy is performed using a gimbal setup that restricts the system to pitch motion only, with additional mass added to the drone to induce instability. The results demonstrate that the LQR controller is capable of stabilising the system. The measured responses exhibit trends similar to those observed in simulation, but with additional oscillations, which are likely caused by modelling errors. The next step for improving controller performance is to carry out system identification experiments for both stationary and free-flight conditions, in order to improve model fidelity and support more accurate controller design.

Acknowledgements

This work was financially supported by the Fonds Wetenschappelijk Onderzoek - Vlaanderen (FWO), grant numbers G021225N, 1SA2U26N

References

[1] Flapper Drones. Flapper nimble+: Bioinspired flying development platform. https://flapper-drones.eu/nimbleplus/, 2026. Accessed: 2026-07-14.

[2] Pakpong Chirarattananon. *Flight Control of a Millimeter-Scale Flapping-Wing Robot*. Phd dissertation, Harvard University, 2014.

[3] Quoc-Viet Nguyen and Woei Leong Chan. Development and flight performance of a biologically-inspired tailless flapping-wing micro air vehicle with wing stroke plane modulation. *Bioinspiration & Biomimetics*, 14(1):016015, 2019.

[4] Zhan Tu, Fan Fei, Jian Zhang, and Xinyan Deng. An at-scale tailless flapping-wing hummingbird robot. i. design, optimization, and experimental validation. *IEEE Transactions on Robotics*, 36(5):1511–1525, 2020.

[5] Hoang Vu Phan, Steven Aurecianus, Taesam Kang, and Hoon Cheol Park. Kubeetle-s: An insect-like, tailless, hover-capable robot that can fly with a low-torque control mechanism. *International Journal of Micro Air Vehicles*, 11:1756829319861371, 2019.

[6] M. Sun. Insect flight dynamics: Stability and control. *Reviews of Modern Physics*, 86(2):615–646, 2014.

[7] D. L. Altshuler, M. Princevac, H. Pan, and J. Lozano. Wake patterns of the wings and tail of hovering hummingbirds. *Experiments in Fluids*, 46:835–846, 2009.

[8] Kai Hu, Huichao Deng, Shengjie Xiao, Gongyu Yang, and Yuhong Sun. Design of a bio-inspired, two-winged, flapping-wing micro air vehicle with high-lift performance. *Journal of Bionic Engineering*, 21(3):1191–1207, 2024.

[9] Hussein Altartouri, A Roshanbin, G Andreolli, L Fazzi, Matej Karásek, Mohamed Lalami, and André Preumont. Passive stability enhancement with sails of a hovering flapping twin-wing robot. *International Journal of Micro Air Vehicles*, 11:1756829319841817, 2019.

http://www.imavs.org/

[10] Zhi Ern Teoh, Sawyer B Fuller, Pakpong Chirarattananon, N. O. Prez-Arancibia, Jack D Greenberg, and Robert J Wood. A hovering flapping-wing microrobot with altitude control and passive upright stability. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 3209–3216. IEEE, 2012.

[11] Shengjie Xiao, Yuhong Sun, Dapeng Ren, Kai Hu, Huichao Deng, Yun Wang, and Xilun Ding. Design and flight performance of a bio-inspired hover-capable flapping-wing micro air vehicle with tail wing. *Aerospace*, 10(11):910, 2023.

[12] Yichen Zhang, Yiming Xiao, Qingcheng Guo, Feng Cui, Jiaxin Zhao, Guangping Wu, Chaofeng Wu, and Wu Liu. Stability and controller research of double-wing fmav system based on controllable tail. *Biomimetics*, 9(8):449, 2024.

[13] T. Roelandt. *Flapping Wing Nano Drones in Hovering Flight: Vehicle Design and Analysis of Flight Performance*. PhD thesis, KU Leuven, Leuven, Belgium, 2025.

[14] S. Timmermans. *Flight Performance of Nano Aerial Vehicles with Flapping Wings: Experiment-Based System Modelling*. PhD thesis, KU Leuven, Leuven, Belgium, 2021.

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

[16] Thomas Roelandt, Arno Melis, Jeroen Vanpoucke, Giheon Ha, Hoon Cheol Park, and Dirk Vandepitte. Comparison of stroke and stroke plane modulation in insect-like flapping wing forward flight control. In *2026 New Trends in Civil Aviation (NTCA)*, pages 15–22. IEEE, 2026.

[17] L. Wang, W. Jiang, L. Zhao, and Z. Jiao. Modeling and hover flight control of a micromechanical flapping-wing aircraft inspired by wing–tail interaction. *IEEE/ASME Transactions on Mechatronics*, 28(6):3132–3142, 2023.

[18] L. Zhao, W. Wang, Y. Chen, and Z. Jiao. Analytical modeling of a hoverable x-shape flapping wing aircraft considering wing–tail interaction. In *Proceedings of the 32nd Congress of the International Council of the Aeronautical Sciences (ICAS)*, pages 1–11, 2021.

[19] M. Karásek. *Robotic hummingbird: Design of a control mechanism for a hovering flapping wing micro air vehicle*. PhD thesis, Universite libre de Bruxelles, Bruxelles, Belgium, 2014.

[20] B. Helsen, T. Roelandt, M. Floren, and D. Vandepitte. Flight control of an insect-like tailless flapping wing nano-robot around hover. In *Proceedings of the 15th Annual International Micro Air Vehicle Conference and Competition*, pages 23–32, 2024.

http://www.imavs.org/

Experimental Identification of the Yaw Damping Derivative of a Flapping-Wing Micro Air Vehicle

D.-J. Liu and W.-L. Chan*
National Cheng Kung University, No. 1, Ta-Hsueh Road, Tainan, Taiwan

http://www.imavs.org/

ABSTRACT

This paper presents an experimental method for identifying the yaw damping derivative of a tailed flapping-wing micro air vehicle (FWMAV). Although flapping-wing vehicles exhibit highly coupled unsteady aerodynamics, experimental characterization of their lateral–directional stability derivatives remains limited. A robotic-arm-based test platform was developed to impose controlled yaw motions under prescribed angles of attack and sideslip conditions while measuring the resulting aerodynamic forces and moments. The platform integrates a six-axis force/torque sensor, optical motion capture, synchronized data acquisition, and a fan-array facility to reproduce a forward-flight inflow condition.

Experimental results show that the yawing moment exhibits a clear and repeatable dependence on yaw rate, enabling identification of the yaw damping derivative, N_r. The identified values remain negative throughout the tested conditions, indicating that the vehicle generates a restoring yawing moment opposing yaw-rate perturbations. In contrast, the side-force response shows no consistent yaw-rate dependence, while the roll moment is governed primarily by sideslip angle rather than yaw rate. These findings provide experimental evidence of passive yaw damping in a tailed FWMAV and demonstrate the feasibility of robotic-arm-based forced-motion testing for identifying lateral–directional stability derivatives to support flight dynamics modeling and controller development.

1 INTRODUCTION

Flapping-wing micro air vehicles (FWMAVs) represent a class of bio-inspired flying machines that generate lift and thrust through reciprocating wing motion. At small scales and low Reynolds numbers, their aerodynamic behavior differs substantially from that of conventional fixed-wing and rotary-wing vehicles. Unsteady mechanisms such as clap-and-fling, leading-edge vortex formation, wake capture, and wing flexibility contribute significantly to their force and moment generation [1, 2, 3, 4]. These mechanisms offer potential advantages in agility and low-speed flight, but they also make the dynamic modeling and control of FWMAVs challenging.

For control-oriented flight dynamics, aerodynamic stability derivatives provide a compact way to describe how forces and moments vary with motion states such as angle of attack, sideslip angle, and angular rates. However, compared with conventional aircraft, the stability derivatives of FWMAVs remain insufficiently characterized. Existing studies have examined insect flight stability, simulation-based FWMAV dynamics, and system identification of flapping-wing vehicles [5, 6, 7, 8]. Among experimental FWMAV studies, the DelFly family is one of the few platforms for which dynamic models have been identified from flight data [9, 10]. Such free-flight identification is valuable because it directly reflects the behavior of the vehicle in flight, but it is naturally limited to flight regimes that are stable, repeatable, and safely achievable.

A complementary approach is to prescribe controlled motion in a laboratory environment while measuring the resulting aerodynamic forces and moments. This idea is closely related to the classical forced-oscillation method used for aircraft stability-derivative identification, in which a model is driven about a specified axis and the aerodynamic response is measured [11]. Robotic-arm-based aerodynamic testing extends this idea by allowing flexible motion trajectories and large attitude excursions. In previous work, a fan-array robotic-arm method was developed for CKopter-1 and used to identify pitch-rate stability derivatives over a broad angle-of-attack range [12]. That study demonstrated that the method can provide useful dynamic-derivative data even in regimes where sustained free flight is not feasible. The present work extends this methodology from longitudinal pitch-rate dynamics to lateral–directional yaw-rate excitation.

In lateral–directional dynamics, yaw damping is a key indicator of directional stability. The yaw-moment derivative with respect to yaw rate, N_r, describes how the yawing moment changes in response to yaw-rate perturbations. A negative N_r indicates that the vehicle generates a yawing moment opposing the yawing motion, thereby providing stabilizing yaw damping. Such damping is particularly important for FWMAVs because their low inertia, unsteady aerodynamics, and multi-axis coupling can make the lateral–directional response difficult to predict from geometry alone. Passive

*Email address: woeileong_chan@gs.ncku.edu.tw

aerodynamic damping has also been discussed in the context of flapping flight and biological flyers, where it contributes to disturbance rejection and flight stability [13, 14].

This paper presents a robotic-arm-based experimental investigation of the yaw damping characteristics of a tailed flapping-wing micro air vehicle (FWMAV). Controlled yaw-sweep motions are imposed under prescribed angles of attack and sideslip conditions while synchronized force/moment and motion measurements are acquired. The primary objective is to experimentally identify the yaw damping derivative, N_r, and to characterize the associated lateral-directional aerodynamic responses. The proposed methodology extends forced-motion testing to flapping-wing MAVs and provides an experimental framework for identifying lateral-directional stability derivatives to support flight dynamics modeling and controller development.

2 Experimental Setup and Methodology

2.1 Flapping-wing MAV platform

The test vehicle used in this study is the tailed flapping-wing micro air vehicle (CKopter-1) previously reported by Chan *et al.* [12]. The vehicle has a total mass of 17.25 g, a wingspan of 200 mm, and employs a pair of reciprocating wings to generate lift and thrust, while a conventional tail provides longitudinal and directional stability and control. A detailed description of the vehicle design, flapping mechanism, structural configuration, and aerodynamic characteristics can be found in [12].

The center of gravity (CG) of the fully assembled FWMAV was determined experimentally by balancing the vehicle on a fine support point until static equilibrium was achieved. The measured CG location was subsequently used to transform the forces and moments measured by the six-axis force sensor from the sensor reference frame to the vehicle CG. A detailed description of the coordinate transformation is available in [12].

To facilitate comparison with the previously identified longitudinal dynamic derivatives, the same experimental configuration and trim condition were adopted throughout this study. The flapping mechanism was operated at the experimentally determined steady-level-flight condition reported in [12], corresponding to a flapping frequency of 14 Hz under a nominal incoming flow of approximately $2\ \mathrm{m/s}$. The elevator and rudder were fixed at their trim positions throughout all experiments so that variations in the measured aerodynamic forces and moments could be attributed primarily to the prescribed yaw motion.

The objective of the present experiments was to identify the yaw damping derivative, N_r, under different angles of attack and sideslip conditions. In addition to the yawing moment, the side force and rolling moment were also examined to assess whether other lateral–directional aerodynamic derivatives associated with yaw-rate excitation could be reliably identified using the proposed robotic-arm-based experimental method.

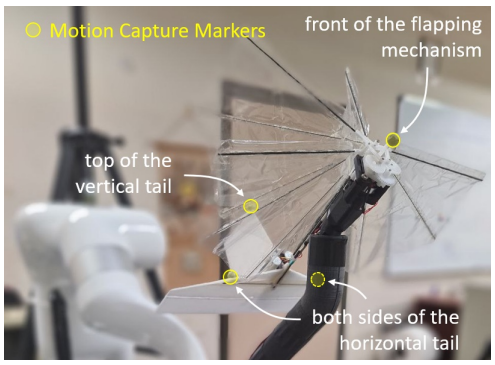


Figure 1: Tailed flapping-wing MAV platform used in this study.

2.2 Robotic-arm-based dynamic test setup

A robotic-arm-based experimental setup was developed to impose repeatable yaw motions on the FWMAV. The vehicle was mounted on a support structure connected to a six-axis force/torque sensor (ATI Nano17 Titanium, ATI Industrial Automation, USA), which measured the three-axis forces and moments throughout the prescribed yaw-sweep motion. The sensor outputs were synchronized with the robotic-arm motion and the optical motion-capture measurements for subsequent aerodynamic derivative identification. This setup extends the fan-array robotic-arm methodology previously developed for pitch-rate dynamic characterization of a flapping-wing MAV to the identification of lateral–directional derivatives under yaw-rate excitation [12].

An optical motion-capture system (Vicon Motion Systems Ltd., UK) was used to measure the actual position and attitude of the FWMAV throughout each experiment. Although the robotic arm prescribed the desired yaw-sweep motion, the commanded trajectory was not used directly for aerodynamic derivative identification. Instead, the motion-capture measurements were used to determine the actual vehicle attitude and yaw rate, thereby accounting for any deviations arising from robot tracking errors, structural compliance, or mounting flexibility. The motion-capture and force/moment measurements were synchronized through a common frame-based data acquisition procedure, enabling the measured aerodynamic loads to be correlated with the instantaneous vehicle motion.

A fan-array flow facility was placed upstream of the model to generate a controlled incoming flow. Since the model was fixed to the robotic-arm platform, the incoming flow was used to reproduce the relative flow condition experienced during forward flight. The incoming flow velocity at the test location was measured using an Extech HD350 differential pressure manometer equipped with a Pitot tube. During fan-array operation, the measured velocity fluctuated about a

http://www.imavs.org/

mean value of approximately 2 m/s. Accordingly, all experiments were conducted under a nominal incoming flow condition of approximately 2 m/s. A detailed characterization of the spatial velocity distribution and turbulence intensity of the fan-array flow was beyond the scope of the present study.

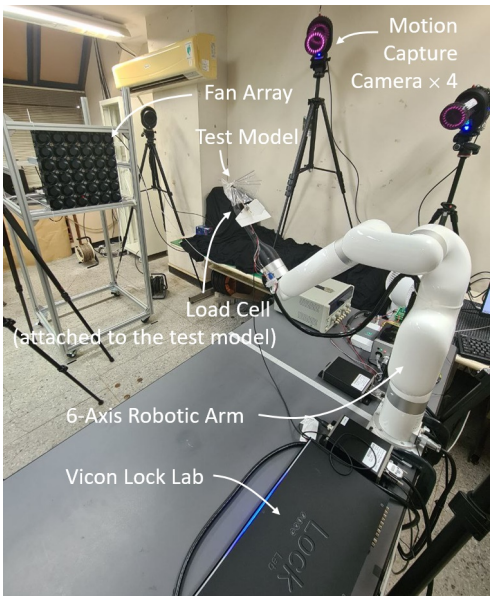


Figure 2: Synchronized robotic-arm-based experimental setup for yaw-rate derivative identification.

2.3 *Test conditions*

The experiments were conducted over a range of angles of attack and sideslip angles. The angle of attack was varied from 30° to 60°, encompassing the experimentally determined steady-level-flight trim condition of approximately 45° reported in [12]. This range was selected to characterize the yaw damping behavior in the vicinity of the nominal operating condition while also examining the influence of moderate deviations in angle of attack. The sideslip angle was varied within the prescribed yaw-sweep range.

For each angle-of-attack condition, the robotic arm executed a programmed yaw sweep from $\beta = -45°$ to $\beta = 45°$ at one of five prescribed sweep speeds, producing five corresponding yaw-rate conditions. To avoid the acceleration and deceleration phases associated with the reversal of the robotic-arm motion, only the data acquired between $\beta = -30°$ and $\beta = 0°$ were used for the subsequent analysis. This portion of the trajectory corresponded to the constant-angular-rate region, ensuring that the identified aerodynamic responses were associated with a well-defined yaw-rate excitation. Each test condition was repeated five times to evaluate experimental repeatability.

The lowest yaw-sweep speed was treated as the quasi-static reference condition, while the higher sweep speeds were used as the dynamic cases. By comparing the measured aerodynamic force and moment responses under different yaw-rate conditions, the yaw damping derivative was identified.

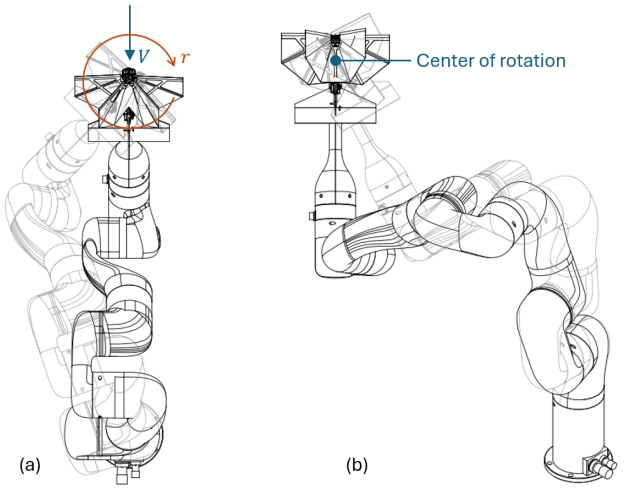


Figure 3: Robotic-arm yaw-sweep motion used to impose controlled sideslip and yaw-rate perturbations: (a) top view showing the incoming flow direction and yaw rate; (b) isometric view showing the center of rotation. The robotic arm executes a programmed yaw sweep from $\beta = -45°$ to $45°$. Only the constant-angular-rate segment between $\beta = -30°$ and $0°$ is included in the data analysis to avoid the acceleration and deceleration phases of the robotic-arm motion.

2.4 *Data processing*

The force/moment and motion-capture data were first synchronized using their corresponding frame indices. The force and moment signals were transformed from the sensor reference frame to the vehicle center of gravity using the experimentally determined CG location, filtered, and resampled to match the motion-capture sampling rate. Bias data collected without incoming flow and without flapping motion were subtracted to remove the static contribution of the mounting structure and sensor offsets.

The sideslip angle, β, was obtained from the attitude measured by the motion-capture system, while the yaw rate, r, was computed by differentiating the measured yaw-angle signal. The measured motion, rather than the commanded robotic-arm trajectory, was used throughout the aerodynamic derivative identification to ensure that the estimated derivatives corresponded to the actual kinematics experienced by the vehicle.

The yaw damping derivative was calculated as

$$N_r = \frac{\partial N}{\partial r}, \tag{1}$$

where N is the yawing moment about the center of gravity and r is the yaw rate. For each angle-of-attack and sideslip-

http://www.imavs.org/

http://www.imavs.org/

angle condition, the variation of yawing moment with yaw rate was used to estimate N_r.

The side-force and roll-moment responses were also examined to assess whether other yaw-rate-related lateral–directional effects could be identified. However, the side-force response did not show a repeatable monotonic trend with increasing yaw rate, and the roll-moment curves were highly overlapped across yaw-sweep speeds. Therefore, the final quantitative identification in this paper focuses on N_r, while the Y and L responses are used qualitatively to interpret the limitations and dominant aerodynamic trends of the experiment.

3 Results and Discussion

3.1 *Lateral–directional force and moment response*

The processed experimental data show that the side force Y, yawing moment N, and rolling moment L respond differently to sideslip angle and yaw-sweep speed. Each test condition was repeated five times. The solid curves in Figs. 4, 5, and 6 represent the mean response of the five runs, while the shaded regions denote ± 1 standard deviation. The observed variability primarily reflects cycle-to-cycle variations in the unsteady flapping-wing aerodynamic loads and measurement uncertainty rather than poor repeatability of the robotic-arm motion.

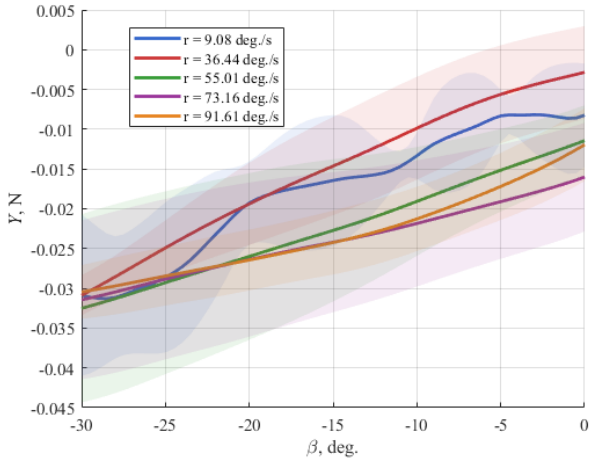


Figure 4: Mean side-force response as a function of sideslip angle at different yaw-sweep speeds for $\alpha = 60°$. Solid lines denote the mean of five repeated experiments, while the shaded regions indicate ± 1 standard deviation. The response does not show a clear monotonic trend with increasing yaw rate.

Figure 4 shows the side-force response at $\alpha = 60°$. Although the side force varies with sideslip angle, the responses obtained at different yaw-sweep speeds do not exhibit a consistent monotonic dependence on yaw rate. The separation among the curves is comparable to the measurement variability, indicating that the yaw-rate contribution to side force is too small to permit reliable identification of the side-force derivative, Y_r.

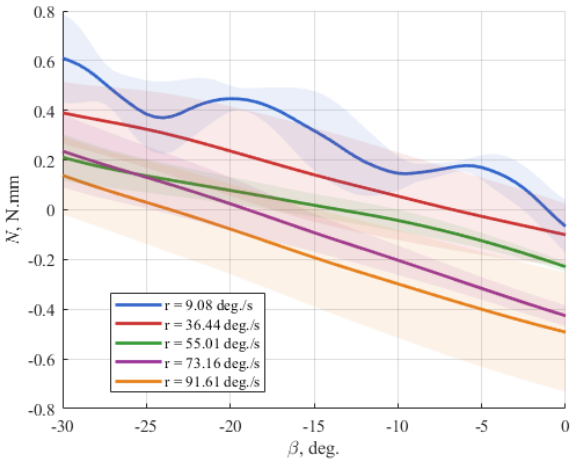


Figure 5: Mean yawing-moment response as a function of sideslip angle at different yaw-sweep speeds for $\alpha = 60°$. Solid lines denote the mean of five repeated experiments, while the shaded regions indicate ± 1 standard deviation. The systematic separation among the curves demonstrates a measurable yaw-rate effect.

In contrast, the yawing-moment response shown in Fig. 5 exhibits a clear and repeatable dependence on yaw-sweep speed. As the yaw rate increases, the yawing-moment curves shift systematically, demonstrating that the yawing moment is sensitive to yaw-rate perturbations. This trend forms the basis for identifying the yaw damping derivative, N_r. Physically, the imposed yaw rate generates an additional lateral velocity at the tail, producing a restoring yawing moment. Consequently, the measured negative values of N_r are consistent with the expected stabilizing directional damping of the tailed FWMAV.

Figure 6 shows that the rolling moment is governed primarily by sideslip angle. The roll moment increases approximately linearly as β changes from $0°$ to $-30°$, while the responses at different yaw-sweep speeds remain highly overlapped. This indicates that the yaw-rate contribution to roll moment is small, suggesting that L_r cannot be reliably identified under the present test conditions. Together with the side-force results, this demonstrates that the dominant measurable yaw-rate effect appears in the yawing moment.

3.2 *Identified yaw damping derivative*

The identified yaw-moment derivative with respect to yaw rate, N_r, remained negative across the tested conditions. This negative sign indicates that the yawing moment generated by the vehicle opposes the imposed yaw-rate perturbation. Therefore, the measured N_r represents a stabilizing yaw-

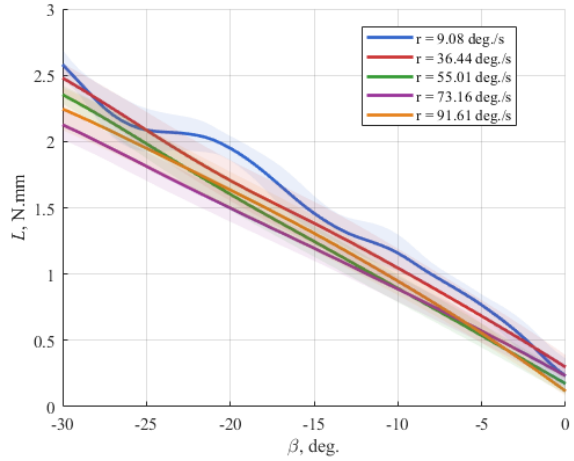


Figure 6: Mean rolling-moment response as a function of sideslip angle at different yaw-sweep speeds for $\alpha = 60°$. Solid lines denote the mean of five repeated experiments, while the shaded regions indicate ± 1 standard deviation. The overlap among the curves suggests that the rolling moment is governed primarily by sideslip angle rather than yaw rate.

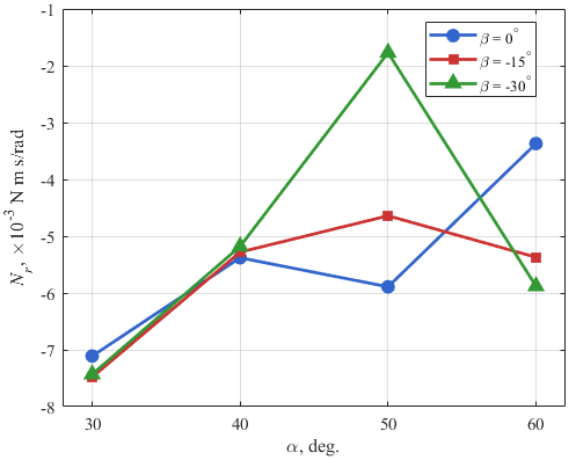


Figure 7: Identified yaw damping derivative, N_r, as a function of angle of attack for the three tested sideslip angles. The negative values indicate a stabilizing yawing moment opposing the imposed yaw-rate perturbation.

damping effect.

The magnitude of N_r varied with angle of attack and sideslip angle, reflecting the influence of the high-angle flapping-wing flow field, tail interaction, and body orientation. Although the variation with angle of attack was not strictly monotonic, N_r remained negative throughout the investigated flight envelope, indicating that the vehicle consistently generated a restoring yawing moment in response to yaw-rate perturbations.

Figure 7 summarizes the identified yaw damping derivative, N_r, as a function of angle of attack for the three tested sideslip angles. The measured N_r values are generally on the order of 10^{-3} N m s/rad.

3.3 Comparison with DelFly II

To assess the physical consistency of the identified yaw damping derivative, the result was compared with the dimensional yaw-rate derivative reported for DelFly II. Although the two platforms differ in geometry, wing configuration, tail arrangement, and test condition, DelFly II provides a useful reference because it is one of the few flapping-wing MAVs with reported lateral– directional aerodynamic derivatives.

Table 1 compares the dimensional yaw damping derivative obtained in this study with that reported for DelFly II. The comparison uses the present data at $\alpha = 60°$ because the DelFly II reference data were obtained at an approximately similar angle of attack. For example, using the reported velocity components $u = 0.25$ m/s and $w = 0.5$ m/s, the corresponding angle of attack is $\alpha = \tan^{-1}(w/u) \approx 63.4°$, which is close to the 60° condition tested in this study.

Table 1: Comparison of dimensional yaw damping derivative between DelFly II [9] and the present FWMAV.

Derivative	DelFly II	Present study
N_r (N m s/rad)	-1.9×10^{-3}	-3.37×10^{-3}

The two values are of the same order of magnitude and have the same negative sign, indicating physically consistent yaw damping behavior. The difference in magnitude may be attributed to differences in vehicle geometry, flight speed, flapping amplitude, tail configuration, and aerodynamic scaling. The present experiment was conducted at a nominal incoming flow velocity of approximately 2 m/s, whereas the DelFly II aerodynamic model was identified from near-hover and slow-forward-flight maneuvers at approximately 0.5 m/s [9]. Nevertheless, the negative N_r value observed in both platforms indicates that yaw-rate perturbations generate a restoring yawing moment, which is consistent with stable lateral–directional damping behavior. Although obtained using different experimental methodologies, the agreement in both sign and order of magnitude provides additional confidence that the identified yaw damping derivative is physically reasonable.

4 Conclusion

This paper presented an experimental method for identifying yaw damping in a tailed flapping-wing micro air vehicle. A synchronized robotic-arm-based test platform was used to impose controlled yaw motions under prescribed angle-of-attack and sideslip conditions. The measured force, moment, attitude, and yaw-rate data were processed to evaluate the lateral–directional aerodynamic response to yaw-rate

http://www.imavs.org/

perturbations.

The results show that the yawing moment exhibits a clear and repeatable dependence on yaw rate, allowing the yaw damping derivative N_r to be identified. The identified N_r remained negative across the tested conditions, indicating that the vehicle generates a yawing moment opposing yaw-rate perturbations. This confirms the presence of stable yaw damping for the present FWMAV configuration.

The side-force response did not show a clear repeatable trend with increasing yaw rate. This suggests that the yaw-rate effect on side force is too small to be reliably captured by the present experimental setup. The roll-moment response showed a strong dependence on sideslip angle but only weak dependence on yaw-sweep speed, indicating that the roll moment is primarily sideslip-dominated rather than yaw-rate-dominated.

These findings provide experimental evidence of yaw damping in a tailed FWMAV and support future lateral–directional dynamic modeling and flight control design. Future work will improve the measurement sensitivity, further characterize the fan-array inflow, extend the experimental envelope, include rudder-input excitation, and incorporate the identified yaw damping derivative into a complete lateral–directional state-space model. In addition, the proposed identification protocol will be evaluated using reference models or simulated datasets with known aerodynamic derivatives to further assess the accuracy and robustness of the estimation method.

ACKNOWLEDGEMENTS

This work was supported by the National Science and Technology Council, Taiwan, under Grant NSTC 114-2221-E-006-063.

REFERENCES

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

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

[3] W. Shyy, M. Berg, and D. Ljungqvist. Flapping and flexible wings for biological and micro air vehicles. *Progress in Aerospace Sciences*, 35:455–505, 1999.

[4] J. M. Birch and M. H. Dickinson. Spanwise flow and the attachment of the leading-edge vortex on insect wings. *Nature*, 412:729–733, 2001.

[5] H. E. Taha, M. R. Hajj, and A. H. Nayfeh. Flight dynamics and control of flapping wing mavs: A review. *Nonlinear Dynamics*, 70:907–939, 2012.

[6] Q. Khan and R. Akmeliawati. Review on system identification and mathematical modeling of flapping-wing micro aerial vehicles. *Applied Sciences*, 11:1546, 2021.

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

[8] M. Y. Bhatti, S.-G. Lee, and J.-H. Han. Dynamic stability and flight control of biomimetic flapping-wing micro air vehicle. *Aerospace*, 8:362, 2021.

[9] J. V. Caetano, C. C. de Visser, G. C. H. E. de Croon, B. Remes, C. de Wagter, J. Verboom, and M. Mulder. Linear aerodynamic model identification of a flapping wing mav based on flight test data. *International Journal of Micro Air Vehicles*, 5(4):273–286, 2013.

[10] S. F. Armanini, M. Karásek, and C. C. de Visser. Global linear parameter-varying modeling of flapping-wing dynamics using flight data. *Journal of Guidance, Control, and Dynamics*, 41:2338–2360, 2018.

[11] D. D. Vicroy. A guide to forced oscillation data processing and analysis. Technical Report NASA/TP–20210023569, NASA, 2023.

[12] W.-L. Chan, D.-J. Liu, H.-Y. Chen, and C.-L. Chin. A fan-array robotic-arm approach to characterization of pitch-rate dynamics of a flapping-wing mav. *Actuators*, 14(12):592, 2025.

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

[14] L. Ristroph, G. Ristroph, S. Morozova, A. J. Bergou, S. Chang, J. Guckenheimer, Z. J. Wang, and I. Cohen. Active and passive stabilization of body pitch in insect flight. *Journal of The Royal Society Interface*, 10(85):20130237, 2013.

http://www.imavs.org/

Bibliography

[1] Joel Klimont, Jakob Buchsteiner, Alexander Lampalzer, and Radu Grosu. Rapid sim-to-real reinforcement learning for autonomous drone racing. In Renaud Kiefer and Sylvain Durand, editors, *17th International Micro Air Vehicle Conference and Competition*, pages 16–23, Strasbourg, France, Sep 2026. Paper no. IMAV2026-1. 11

[2] Dequan Ou, Quentin Missinne, Christophe De Wagter, and Guido de Croon. Precision home landing for bee-nav robot navigation. In Renaud Kiefer and Sylvain Durand, editors, *17th International Micro Air Vehicle Conference and Competition*, pages 24–31, Strasbourg, France, Sep 2026. Paper no. IMAV2026-2. 11, 12

[3] Marco Tayar, Felipe Tommaselli, Gianluca Capezzuto, Pedro Antinio Rabelo Saraiva, Pedro de Freitas, Lucas Kido, Guilherme Sonego, Ricardo Godoy, and Marcelo Becker. Vln on the fly: An onboard vision-language navigation stack for aerial robots. In Renaud Kiefer and Sylvain Durand, editors, *17th International Micro Air Vehicle Conference and Competition*, pages 32–38, Strasbourg, France, Sep 2026. Paper no. IMAV2026-3. 11

[4] Ishaq Hafez and Karl Stol. Mrac-based admittance control for aerial physical interaction. In Renaud Kiefer and Sylvain Durand, editors, *17th International Micro Air Vehicle Conference and Competition*, pages 39–46, Strasbourg, France, Sep 2026. Paper no. IMAV2026-4. 11

[5] Pedro Souza, Karl Stol, and Ishaq Hafez. Variable admittance control for obstacle separation in aerial manipulation. In Renaud Kiefer and Sylvain Durand, editors, *17th International Micro Air Vehicle Conference and Competition*, pages 47–54, Strasbourg, France, Sep 2026. Paper no. IMAV2026-5. 11, 12

[6] Jakob Harzer, Felix Renard, Kim Polkläser, Dieter Moormann, and Moritz Diehl. Online wind field estimation for airborne wind energy systems in circular flight. In Renaud Kiefer and Sylvain Durand, editors, *17th International Micro Air Vehicle Conference and Competition*, pages 55–62, Strasbourg, France, Sep 2026. Paper no. IMAV2026-6. 11, 12

[7] Aurélie Kempf, Emmanuel Roussel, Guillaume Strub, Jules Farnault, Damien Delmas, and Edouard Laroche. Sequential estimation of fixed-wing uav aerodynamic coefficients: Evaluation on pitch/roll flight data. In Renaud Kiefer and Sylvain Durand, editors, *17th International Micro Air Vehicle Conference and Competition*, pages 63–70, Strasbourg, France, Sep 2026. Paper no. IMAV2026-7. 11

[8] Jasper Lee, Jean-Marc Ingargiola, and Stéphane Viollet. Trimorph: The transformative tricopter. In Renaud Kiefer and Sylvain Durand, editors, *17th International Micro Air Vehicle Conference and Competition*, pages 71–77, Strasbourg, France, Sep 2026. Paper no. IMAV2026-8. 11

[9] Joel Klimont, Jakob Buchsteiner, Alexander Lampalzer, and Radu Grosu. Weitflug: A research-oriented fork of betaflight for agile autonomous flight. In Renaud Kiefer and Sylvain Durand, editors, *17th International Micro Air Vehicle Conference and Competition*, pages 78–84, Strasbourg, France, Sep 2026. Paper no. IMAV2026-9. 11, 12

[10] Alvaro Gaona, David Perez-Saura, and Pascual Campoy Francisco Anguita. Flight-ready lidar-inertial odometry for embedded drone platforms. In Renaud Kiefer and Sylvain Durand, editors, *17th International Micro Air Vehicle Conference and Competition*, pages 85–91, Strasbourg, France, Sep 2026. Paper no. IMAV2026-10. 11

[11] João Machado, Zeynep Bilgin, Matthieu Verdoucq, and Murat Bronz. Lidarflow: Real-time panel-based mav guidance in unknown environments. In Renaud Kiefer and Sylvain Durand, editors, *17th International Micro Air Vehicle Conference and Competition*, pages 92–99, Strasbourg, France, Sep 2026. Paper no. IMAV2026-11. 11

[12] Heitor Riquelme, João Melo, João Pedro Mafalado, Pedro César, Driele Pires, Mateus Ferreira, Bruno Eduardo, Edna Barros, and Paulo Rosa. Monocular geo-localization of terrestrial objects from uav imagery with spatial refinement. In Renaud Kiefer and Sylvain Durand, editors, *17th International Micro Air Vehicle Conference and Competition*, pages 100–107, Strasbourg, France, Sep 2026. Paper no. IMAV2026-12. 11

http://www.imavs.org/

[13] Tomaso De Ponti, Ewoud Smeur, and Bart Remes. Actuator nonlinear fault-tolerant control of a fast-spinning quadrotor with loss of two opposing rotors. In Renaud Kiefer and Sylvain Durand, editors, *17^{th} International Micro Air Vehicle Conference and Competition*, pages 108–117, Strasbourg, France, Sep 2026. Paper no. IMAV2026-13. 11, 12

[14] Daniel Engelsman and Itzik Klein. Aero-lqg: Aerial-enabled robust optimization for lqg-based quadrotor flight controller. In Renaud Kiefer and Sylvain Durand, editors, *17^{th} International Micro Air Vehicle Conference and Competition*, pages 118–125, Strasbourg, France, Sep 2026. Paper no. IMAV2026-14. 11

[15] Salim Al-zubaidi, Jaap Skinner, and Karl Stol. Parameter sensitivity in control allocation: A monte carlo analysis for an over-actuated aerial robot. In Renaud Kiefer and Sylvain Durand, editors, *17^{th} International Micro Air Vehicle Conference and Competition*, pages 126–133, Strasbourg, France, Sep 2026. Paper no. IMAV2026-15. 11

[16] Gaétan Pérez, Lou Denis, Hamza Chakraa, and Murat Bronz. A multi-view, multi-lidar synchronised setup for smoke plume data acquisition: A simulated drone swarm observation case study. In Renaud Kiefer and Sylvain Durand, editors, *17^{th} International Micro Air Vehicle Conference and Competition*, pages 134–141, Strasbourg, France, Sep 2026. Paper no. IMAV2026-16. 11

[17] Yufei Peng, Ni Li, and Hann Woei Ho. Vision-guided velocity control of micro air vehicles using rate-decoded spiking neural network. In Renaud Kiefer and Sylvain Durand, editors, *17^{th} International Micro Air Vehicle Conference and Competition*, pages 142–149, Strasbourg, France, Sep 2026. Paper no. IMAV2026-17. 11

[18] Hamza Chakraa, Maël Feurgard, and Murat Bronz. Decentralized pso-based guidance for multi-uav wildfire monitoring and frontier tracking. In Renaud Kiefer and Sylvain Durand, editors, *17^{th} International Micro Air Vehicle Conference and Competition*, pages 150–157, Strasbourg, France, Sep 2026. Paper no. IMAV2026-18. 11

[19] Saleh Abuali, Natalio Mendoza-Robles, Bogdan Robu, Nicolas Marchand, Sylvain Durand, and Lara Briñón-Arranz. Distributed formation control based on bounded voronoi and cable tension for cooperative hovering of a cable-suspended load. In Renaud Kiefer and Sylvain Durand, editors, *17^{th} International Micro Air Vehicle Conference and Competition*, pages 158–165, Strasbourg, France, Sep 2026. Paper no. IMAV2026-19. 11

[20] Abner Asignacion Jr and Satoshi Suzuki. A structure-aware reactive motion planner for autonomous nano-uav navigation in confined 3d tunnels using minimal range sensing. In Renaud Kiefer and Sylvain Durand, editors, *17^{th} International Micro Air Vehicle Conference and Competition*, pages 166–172, Strasbourg, France, Sep 2026. Paper no. IMAV2026-20. 11, 12

[21] Andreas Zwanenburg and Salua Hamaza. Downscaling autonomous navigation of tiny drones under dense forest canopies. In Renaud Kiefer and Sylvain Durand, editors, *17^{th} International Micro Air Vehicle Conference and Competition*, pages 173–180, Strasbourg, France, Sep 2026. Paper no. IMAV2026-21. 11, 12

[22] Maël Feurgard, Gautier Hattenberger, Nicolas Durand, and Simon Lacroix. Safe synchronous path planning for a fleet of fixed-wing constant speed aircraft. In Renaud Kiefer and Sylvain Durand, editors, *17^{th} International Micro Air Vehicle Conference and Competition*, pages 181–188, Strasbourg, France, Sep 2026. Paper no. IMAV2026-22. 11

[23] Ivan Guerra Bello, Austin Lopez, and Mauro Gallo. Center-of-gravity placement effects on a bio-inspired flapping-wing mav: An empirical characterization for sensor payload integration. In Renaud Kiefer and Sylvain Durand, editors, *17^{th} International Micro Air Vehicle Conference and Competition*, pages 189–194, Strasbourg, France, Sep 2026. Paper no. IMAV2026-23. 11

[24] Arno Melis, Jeroen Vanpoucke, Thomas Roelandt, Jan Swevers, and Dirk Vandepitte. Pitch stabilisation and longitudinal control of a hummingbird-inspired drone using an active tail. In Renaud Kiefer and Sylvain Durand, editors, *17^{th} International Micro Air Vehicle Conference and Competition*, pages 195–202, Strasbourg, France, Sep 2026. Paper no. IMAV2026-24. 11

[25] De-Jing Liu and Woei-Leong Chan. Experimental identification of the yaw damping derivative of a flapping-wing micro air vehicle. In Renaud Kiefer and Sylvain Durand, editors, *17^{th} International Micro Air Vehicle Conference and Competition*, pages 203–208, Strasbourg, France, Sep 2026. Paper no. IMAV2026-25. 11

http://www.imavs.org/