

http://www.imavs.org/

IMAV 2025

16th International Micro Air Vehicle Conference and Competition

November 3-7, 2025

San Andrés Cholula, Puebla, Mexico

www.imavs.org

16TH INTERNATIONAL MICRO AIR VEHICLE CONFERENCE AND COMPETITION

Contents

IMAV2025 **1**
Preface . . . 3
International IMAV Committee . . . 4
Organizing Committee . . . 4
Sponsors . . . 5
Call for Papers . . . 6
Sessions . . . 7
Keynote Speakers . . . 8
List Of Papers . . . 14
Authors . . . 15
Reviewers . . . 16

Papers **17**
3D CNN and Multi-Head Attention to Relative Camera Position . . . 18
A Review of Visual SLAM Systems and Single-Image Depth Estimation for Challenging Scenarios . . . 25
Synthetic Dataset Generation for Object Detection in MAV Applications . . . 32
Intelligent Vision-Based Quadrotor Path Follower . . . 37
Autonomous UAV Landing with Visual Servoing and Semi-Markov Decision Processes . . . 43
Model-Based Analysis of UAV Accurate Landing in Stochastic Turbulent Environments . . . 51
Reinforcement Learning-based Optimal Guidance for Landing the Variable Skew Quad Plane on a Ship . . . 59
A LiDAR-Based Deep Reinforcement Learning Autonomous Navigation System in Unknown Environments . . . 69
Radio Coverage-Aware UAV Planning and Navigation in Urban Scenarios . . . 76
ADRC-Driven Trajectory Control of a Quadcopter Considering Ground Effect Dynamics . . . 82
CU-Fundamental: A Comprehensive Simulation Framework for Autonomous MAV . . . 90
A Comparative Study on Thrust Map Estimation for Multirotor Aerial Vehicles . . . 98
A Comparison of Optimised, Fixed-Tilt, Fully-Actuated Multirotor UAV Configurations . . . 106
A Framework for Indoor-Monocular Depth Estimation on UAVs Using a Thermal Camera . . . 112
Energy-Aware Hybrid Event-Triggered Control for Micro-Drones . . . 118
Payload State Estimation for Cooperative Manipulation Using Multiple UAVs . . . 125
Spiking Neural Networks for High-Speed Continuous Quadcopter Control Using Proximal Policy... . . . 133
Design of Frontal Perching Mechanism to Vertical Surfaces for Flapping-Wing MAVs . . . 142
Pecking Mechanism Inspired by Avian Cranial Kinesis for Flapping-Wing Aerial Manipulators . . . 148
Flight Pattern and Experimental Flights of a Quadricopter Drone at the Edge of a Smoke Plume . . . 156
Dielectric Barrier Discharge Plasma Actuators to Mitigate Flow Separation . . . 162

http://www.imavs.org/

Preface

On behalf of the Local Organizing Committee, it is our pleasure to present the proceedings of the 16^{th} International Micro Air Vehicle Conference and Competition, which was held in San Andrés Cholula, Puebla, Mexico from November 3-7, 2025. For the first time, the IMAV had a physical event in Mexico.

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-one peer-reviewed scientific papers by seventy authors organized in four sessions presented at the IMAV in 2025. The topics of these papers contain a nice mix ranging from aerial vehicle design and energy sources to control, navigation and perception. Together, the papers give an overview of the current state-of-the-art in the field of Micro Air Vehicles. Based on the quality of the scientific and technical contributions, papers were selected to be published in two scientific journals: the International Journal of Micro Air Vehicles (Sage), and Unmanned Systems (World Scientific).

We would like to express our sincere gratitude for the guidance and support of the members of the seventeen members of the International Committee from thirteen institutes who guide IMAV overall, and the eighteen reviewers from seven institutes, who played an important role in assuring the quality of all the papers. In addition to the presentation of the scientific papers, keynote talks were delivered by experts in the field, whom we want to thank for their valuable contributions. We also want to thank all members of the Local Organizing Committee for their invaluable support in the organization of this IMAV-2025.

Last but not least, we extend our deepest gratitude to Christophe De Wagter for his invaluable support each year in the edition and production of the IMAV proceedings.

San Andrés Cholula, Puebla, Mexico. November 2025

Jose Martinez-Carranza
General Chair
email: carranza@inaoep.mx

http://www.imavs.org/

International IMAV Committee

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

Organizing Committee

Jose Martinez-Carranza (INAOE) . . . . . General Chair
Z. Jezabel Guzmán Zavaleta (UDLAP) . . Conference Chair
Hugo Rodríguez-Cortés (CINVESTAV) . . . . . Indoors Chair
Cesar Martínez Torres (UDLAP) . . . . Indoors and Sponsors Chair
Caleb Rascón Estebané (UNAM) . . . . . Indoors Chair
Marco Antonio Negrete Villanueva (UNAM) Indoors Chair
Aldrich A. Cabrera Ponce (BUAP) . . . . . Indoors Chair
Gustavo Rodríguez Gómez (INAOE) . . . . . Indoors Chair
Dr. J. Fermi Guerrero Castellanos (BUAP) . Outdoors Chair
Dr. Diego Mercado Ravell (CINVESTAV) . Outdoors Chair
Dr. Hernán Abaunza González (ITESM) . . Outdoors Chair
Dra. L. Oyuki Rojas Pérez (INAOE) . . . . . Outdoors Chair
Dr. Alejandro Gutierrez Giles (INAOE) . . . Outdoors Chair

http://www.imavs.org/

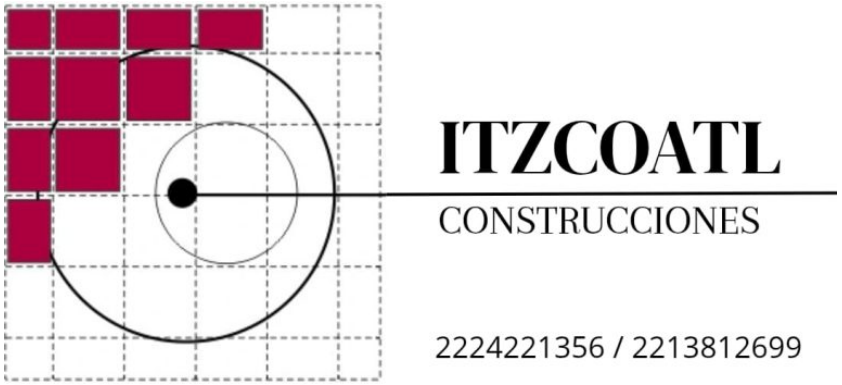
Sponsors











http://www.imavs.org/

Call for Papers



http://www.imavs.org/

The International Micro Air Vehicle Conference and Competitions (IMAV) is a yearly event that aims at fostering key technologies for the development of micro-air vehicles. Since early editions, the IMAV has offered a pioneer event format where competitions are carried out alongside a two-day conference. Highly recognized members of the community are invited to deliver keynotes, while academics are invited to submit research and technical papers. After a peer-reviewed process, accepted papers are presented during the conference.

This edition of the IMAV will be held in the Universidad de las Américas Puebla (UDLAP) located in the city of San Andres Cholula, in the State of Puebla, México. The conference will take place from the 3rd to 7th of November, 2025.

Call for papers

The call for papers for IMAV 2025 welcomed contributions to all aspects of MAVs, including, but not limited to:

Topics for scientific and technical papers include, but are not limited to:

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

Dedicated applicative sessions will be set up for the following topics:

- Atmospheric research
- Archaeological research
- Search and rescue operations
- Industrial inspection
- Agriculture & environment
- Artificial Intelligence for MAVs
- Ethics & Regulations
- Societal impact of MAVs

Based on the quality of the papers and after a thorough evaluation by the IMAV's international committee, selected papers and finalist to the "Best Conference Paper Award" will be offered to be published in one of the following two Journals: International Journal of Micro Air Vehicles and in the Unmanned Systems.

Venue

- Location: San Andrés Cholula, Puebla, Mexico
- This year, the IMAV 2025 conference will be hosted at UDLAP's Auditorium Guillermo and Sofia Jenkinns.
- Event dates: 3-7 November 2025.
- Web site: http://www.imavs.org/2025

Conference Oral Presentations

Session 1: Vision-Based Perception

3D CNN and Multi-Head Attention to Relative Camera Position . . . p.18
A Review of Visual SLAM Systems and Single-Image Depth Estimation for Challenging Scenarios . . . p.25
Synthetic Dataset Generation for Object Detection in MAV Applications . . . p.32

Session 2: Navigation, Guidance, and Path Planning

Intelligent Vision-Based Quadrotor Path Follower . . . p.37
Autonomous UAV Landing with Visual Servoing and Semi-Markov Decision Processes . . . p.43
Model-Based Analysis of UAV Accurate Landing in Stochastic Turbulent Environments . . . p.51
Reinforcement Learning-based Optimal Guidance for Landing the Variable Skew Quad Plane on a Ship . . . p.59
A LiDAR-Based Deep Reinforcement Learning Autonomous Navigation System in Unknown Environments . . . p.69
Radio Coverage-Aware UAV Planning and Navigation in Urban Scenarios . . . p.76
ADRC-Driven Trajectory Control of a Quadcopter Considering Ground Effect Dynamics . . . p.82

Session 3: Control and Estimation

CU-Fundamental: A Comprehensive Simulation Framework for Autonomous MAV . . . p.90
A Comparative Study on Thrust Map Estimation for Multirotor Aerial Vehicles . . . p.98
A Comparison of Optimised, Fixed-Tilt, Fully-Actuated Multirotor UAV Configurations . . . p.106
A Framework for Indoor-Monocular Depth Estimation on UAVs Using a Thermal Camera . . . p.112
Energy-Aware Hybrid Event-Triggered Control for Micro-Drones . . . p.118
Payload State Estimation for Cooperative Manipulation Using Multiple UAVs . . . p.125

Session 4: Bio-Inspired, Morphing Aerial Systems, and Aerodynamics

Spiking Neural Networks for High-Speed Continuous Quadcopter Control Using Proximal Policy... . . . p.133
Design of Frontal Perching Mechanism to Vertical Surfaces for Flapping-Wing MAVs . . . p.142
Pecking Mechanism Inspired by Avian Cranial Kinesis for Flapping-Wing Aerial Manipulators . . . p.148
Flight Pattern and Experimental Flights of a Quadricopter Drone at the Edge of a Smoke Plume . . . p.156
Dielectric Barrier Discharge Plasma Actuators to Mitigate Flow Separation . . . p.162

http://www.imavs.org/

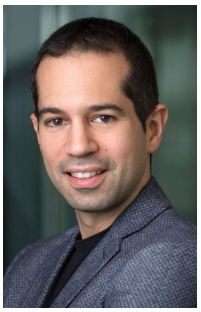
Keynote Speakers

Agile Vision-based Flight

Prof. Davide Scaramuzza - Prof. of Robotics and Perception at the University of Zurich

Abstract

Autonomous drones play a crucial role in inspection, agriculture, logistics, and search-and-rescue missions and promise to increase productivity by a factor of 10. However, they still lag behind human pilots in speed, versatility, and robustness. What does it take to fly autonomous drones as agile as or even better than human pilots? Autonomous, agile navigation through unknown, GPS-denied environments poses several challenges for robotics research regarding perception, learning, planning, and control. In this talk, I will show how the combination of model-based and machine-learning methods, united with the power of new, low-latency sensors, such as event cameras, can allow drones to achieve unprecedented speed and robustness by relying solely on onboard computing. This can result in better productivity and safety of future autonomous aircraft.



Biography

Davide Scaramuzza is a Professor of Robotics and Perception at the University of Zurich. He did his Ph.D. at ETH Zurich, a postdoc at the University of Pennsylvania, and was a visiting professor at Stanford University. His research focuses on autonomous, agile microdrone navigation using standard and event-based cameras. He pioneered autonomous, vision-based navigation of drones, which inspired the navigation algorithm of the NASA Mars helicopter and many drone companies. He contributed significantly to visual-inertial state estimation, vision-based agile navigation of microdrones, and low-latency, robust perception with event cameras, which were transferred to many products, from drones to automobiles, cameras, AR/VR headsets, and mobile devices. In 2022, his team demonstrated that an AI-powered drone could outperform the world champions of drone racing, a result published in Nature and considered the first time an AI defeated a human in the physical world. He is a consultant for the United Nations on disaster response and disarmament. He has won many awards, including an IEEE Technical Field Award, the levation to IEEE Fellow, the IEEE Robotics and Automation Society Early Career Award, a European Research Council Consolidator Grant, a Google Research Award, two NASA TechBrief Awards, and many paper awards. In 2015, he co-founded Zurich-Eye, today Meta Zurich, which developed the world-leading virtual-reality headset Meta Quest. In 2020, he co-founded SUIND, which builds autonomous drones for precision agriculture. Many aspects of his research have been featured in the media, such as The New York Times, The Economist, and Forbes.

http://www.imavs.org/

Neuromorphic Artificial Intelligence for small, autonomous drones

Prof. Guido De Croon - Prof. in Bio-inspired Micro Air Vehicles Delft University of Technology

Abstract

Small, autonomous drones are promising for many applications, such as search-and-rescue, greenhouse monitoring, or keeping track of stock in warehouses. Since they are small, they can fly in narrow areas. Moreover, their light weight makes them very safe for flight around humans. However, making small drones fly completely by themselves is an enormous challenge. Most approaches to Artificial Intelligence for robotics have been designed with self-driving cars or other large robots in mind – and these are able to carry many sensors and ample processing. In my talk, I will argue that a different approach is necessary for achieving autonomous flight with small drones. In particular, I will discuss how we can draw inspiration from flying insects and endow our drones with similar intelligence. Examples include flapping-wing drones that can deal with collisions, and swarms of CrazyFlie quadrotors of 30 grams able to explore unknown environments and find gas leaks. The intelligence of these small drones will be further enhanced by novel neuromorphic sensing and processing technologies. I will discuss recent experiments we performed on neuromorphic AI in our lab – from ego-motion estimation to agile flight. Finally, I will present our research in the context of the European SPEAR project: that it is possible to evolve drone bodies for specific tasks, outcompeting traditional drone designs in tasks such as drone racing.



Biography

Received his M.Sc. and Ph.D. in the field of Artificial Intelligence (AI) at Maastricht University, the Netherlands. His research interest lies in computationally efficient and often bio-inspired algorithms for robot autonomy, with an emphasis on computer vision. Since 2008, he has worked on algorithms for achieving autonomous flight with small and lightweight flying robots, such as the DelFly flapping wing MAV. In 2011-2012, he was a research fellow in the Advanced Concepts Team of the European Space Agency, where he studied topics such as optical flow-based control algorithms for extraterrestrial landing scenarios. Currently, he is an associate professor at TU Delft and scientific lead of the Micro Air Vehicle lab (MAV-lab) of Delft University of Technology.

http://www.imavs.org/

A Quest for Autonomously Exploring Drones

Prof. Hyunchul Shim - Prof. in the Department of Aerospace Engineering at Korea Advanced Institute of Science and Technology

Abstract

Towards the end of the 20th century, a Japanese TV show featured electric helicopters flying through complex mazes, hand-controlled by human pilots for a big prize. Watching it, I wondered: what if UAVs could do this autonomously? At that time, drones were bulky—typical drones in those days were the size of a motorbike—and onboard SLAM or lightweight perception was still far in the future. By the mid-2010s, the landscape had shifted: small multirotors had emerged, and GPUs became compact enough to fly. In 2016, I co-founded what is recognized as the first autonomous drone racing competition, held at IROS in Daejeon. Since then, we have witnessed a surge of autonomous drone challenges, from the DARPA Subterranean Challenge to Lockheed Martin's AlphaPilot and, most recently, the UAE's A2RL DCL. In this talk, I will reflect on this trajectory—from the point when quadrotors came to symbolize drones, through the breakthroughs enabled by global competitions, to the lessons learned from my own first-hand experiences. I will also discuss the challenges ahead in building more capable, trustworthy, and socially beneficial drone technologies.



Biography

Hyunchul Shim received the B.S. and M.S. degrees in mechanical design and production engineering from Seoul National University, Seoul, Korea, in 1991 and 1993, respectively, and the Ph.D. degree in mechanical engineering from the University of California Berkeley, Berkeley, USA in 2000. From 1993 to 1994, he was with Hyundai Motor Company, Korea. From 2001 to 2005, he was with Maxtor Corporation, Milpitas, CA, USA as Staff Engineer. From 2005 to 2007, he was with the University of California Berkeley as Principal Engineer, in charge of Berkeley Aerobot Team. In 2007, he joined the Department of Aerospace Engineering, KAIST, South Korea, as an Assistant Professor. He is now Professor in School of Electrical Engineering. During his career at KAIST, he has led a number of efforts as Director of KI Robotics Institute at KAIST from '19-'22, advisor for RPAS panel at ICAO, and Director of Korea Civil RPAS Research Center. He is also very active in a number of world's premier competitions such as DARPA SubT, Lockheed Martin AlphaPilot, Indy Autonomous Challenges. He has received a number of major awards from Korean government and global events. His interests center on the combination of robotics and AI with aerial and ground vehicles for highly intelligent vehicles.

http://www.imavs.org/

Detection and Classification of Small UAVs

Capt. Rafael Cruz Gómez and Lt. Gustavo Guerrero Clavel - Captain Rafael Alberto Cruz Gómez and Lieutenant Gustavo Guerrero Clavel of the Secretariat of the Navy (SEMAR)

Abstract

Small Unmanned Aerial Vehicles (UAV) or drones are providing big amounts of data. Often, new classes of drones are deployed on a regular basis to help in different tasks. The purpose of this talk is to present a technique of drone detection using acoustic means. Although there are several techniques of drone detection, such as electromagnetic signal detection, radar location, electro-optic, and infrared visual detection. It is feasible to detect the unique acoustic signature of a drone class using modern techniques of signal processing. Using acoustic sensors, a drone can be detected, and with machine learning techniques, its path can be tracked. Furthermore, there are several other data that can be extracted using deep learning techniques based on neural networks such as the class of the drone, which can help in the decision process of the main function of the drone. At the end of the talk is presented a small routine used to detect and classify drones using its acoustic signature.

Biography

Captain Gómez graduated as an electrical engineer from the Escuela de Ingenieros of the Mexican Navy. He has a dual master's degree in Master of Sciences of Electrical Engineering and Master of Sciences in Engineering Acoustics from the United States Naval Postgraduate School. He has held several appointments in the Mexican Navy such as the Embedded Systems designer for UAV SPARTAAMv44, SPARTAAMv88 and, SPARTAAMv200. LCDR Gómez has been awarded with several distinctions, including Medals of Perseverance, Mexican Navy Research, and Development distinctive, and Honorable Mention for the excellence in the design of UAV. Currently, LCDR Gómez is the head of the Embedded Systems Lab in the SPARTAAM UAS Program at the Mexican Navy.

Lieutenant Guerrero, Electronics and Naval Communications Engineer, graduated from the Naval Engineers School of Mexico (2006–2011) and completed a specialisation in Operational Logistics at CESNAV in 2021. He has received technical training from Microchip, a PCB Design Course for military, aerospace, and extreme environments, and IPC/WHMA-A-620 certification from the Institute of Printed Circuits. He has served in various positions at INIDETAM and UNINDETEC, and is currently Head of the Integration and Technical Support Laboratory for Unmanned Aircraft. His work includes avionics engineering in the design, construction, and testing of avionics subsystems, simulation models, UAV prototype integration, as well as operation and technical support for UAV systems such as SPARTAAM-44, SPARTAAM-88, SPARTAAM-VTOL, and SPARTAAM-Satellite. For his contributions, he was awarded the Technological Research and Development Distinction.

http://www.imavs.org/

MACE: Development of a Medium-Altitude Fixed-Wing Drone for Environmental Monitoring

Prof. Tom Richardson Professor Tom Richardson, Faculty of Engineering, University of Bristol, UK.

Abstract

Natural events, particularly volcanic eruptions, release tiny particles (aerosols) into the atmosphere and offer invaluable real-world opportunities to study processes relevant to climate science. However, safely and rapidly collecting data from these events is challenging. The MACE project is developing advanced, automated drone technology specifically designed for observing and analysing emissions from active volcanoes. These drones will operate at altitudes up to 10km above sea level and will be flown over regularly erupting volcanoes ,including Volcán de Fuego (Guatemala), Soufrière Hills (Montserrat), and Lascar (Chile). By analysing these natural volcanic emissions in situ, the research will investigate how tiny cloud droplets form and how natural aerosol layers affect radiation. A key goal is to develop a rapid-response capability using these drones, enabling the scientific community to safely gather crucial data from future significant volcanic eruptions, thereby improving our understanding of natural climate processes.



Biography

Tom Richardson is Professor of Aerial Robotics at the University of Bristol in the UK and specializes in the application of modern control theory and novel sensors to UAS/Drones. He has been granted permission for BVLOS (Beyond Visual Line of Sight) operations in multiple countries and was the International Drone Safety Lead for the Multinational Deep Carbon Observatory (DCO) funded ABOVE field campaign to Papua New Guinea in 2019. Tom is a founding partner of Perceptual Robotics, has held an NPPL (pilots license) for over 15 years and has worked with a range of industrial partners including DSTL, BAE Systems and Thales. Recent projects include MIMRee on the robotic inspection and maintenance of wind turbines and Prometheus on the use of drones for underground inspection.

http://www.imavs.org/

The history of a drone show company

Xavier Páez Casanovas, WG Drones Company

Abstract

WG Drones is a company specializing in the design, manufacture, and sale of drones specifically for light shows and aerial displays. The company was founded in 2019 and is based in Puebla, Mexico, with an operational base in Spain. Its core business revolves around creating drones equipped with bright 2200 Lumens RGBW LEDs that are compatible with professional drone show software, catering to the growing market for dynamic aerial entertainment.



The company serves a global clientele, providing not only the hardware but also essential equipment and training for drone show providers. WG Drones has successfully sold its products and established a presence in a diverse range of countries across multiple continents, including Argentina, Australia, Chile, Colombia, the Dominican Republic, India, Italy, Qatar, the Maldives, Mexico, New Zealand, South Korea, Spain, and the United States. Notable projects include the use of their DR-001 model by Citelum Mexico for a Valentine's Day show in Puebla, and the deployment of 1,000 WG Drones by Flock Drone Art for the opening ceremony of the America's Cup 2024 in Barcelona, Spain.

In 2024, in a significant move to strengthen its position in the European market, WG Drones has expanded its operations in Spain. This expansion aims to meet the increasing demand for advanced drone show technology throughout Europe. While specific details regarding a large-scale manufacturing facility are not explicitly public, the establishment of the Spanish entity suggests a focused effort to provide local support, sales, and potentially service options for its European clientele.

Biography

Xavier Páez is the founder and current CEO of WG Drones, a company that blends creativity, engineering, and cutting-edge technology to design and execute aerial light shows and multimedia experiences. With over 14 years of experience in water feature design and technological innovation, Xavier has led international projects that integrate art, engineering, and immersive technology. He began his entrepreneurial journey as the founder of WG Water Gush, a company recognized for pioneering the concept of the "Social Media Fountain", combining interactive lighting and fountain systems for public and private spaces. In 2023, Xavier expanded his vision by creating WG Drones, taking visual experiences beyond the water and into the sky through advanced drone light show technology. In 2024, WG Drones established a new base in Girona, Spain, continuing its mission to redefine outdoor entertainment through synchronized drone performances and large-scale multimedia events. Xavier's leadership is guided by a strong belief in technological development, strategic innovation, and creative excellence, driving WG Drones to the forefront of international event technology.

http://www.imavs.org/

Papers

[1] 3D CNN and Multi-Head Attention to Relative Camera Position p.18
[2] A Review of Visual SLAM Systems and Single-Image Depth Estimation for Challenging Scenarios p.25
[3] Synthetic Dataset Generation for Object Detection in MAV Applications p.32
[4] Intelligent Vision-Based Quadrotor Path Follower p.37
[5] Autonomous UAV Landing with Visual Servoing and Semi-Markov Decision Processes p.43
[6] Model-Based Analysis of UAV Accurate Landing in Stochastic Turbulent Environments p.51
[7] Reinforcement Learning-based Optimal Guidance for Landing the Variable Skew Quad Plane on a Ship p.59
[8] A LiDAR-Based Deep Reinforcement Learning Autonomous Navigation System in Unknown Environments p.69
[9] Radio Coverage-Aware UAV Planning and Navigation in Urban Scenarios p.76
[10] ADRC-Driven Trajectory Control of a Quadcopter Considering Ground Effect Dynamics p.82
[11] CU-Fundamental: A Comprehensive Simulation Framework for Autonomous MAV p.90
[12] A Comparative Study on Thrust Map Estimation for Multirotor Aerial Vehicles p.98
[13] A Comparison of Optimised, Fixed-Tilt, Fully-Actuated Multirotor UAV Configurations p.106
[14] A Framework for Indoor-Monocular Depth Estimation on UAVs Using a Thermal Camera p.112
[15] Energy-Aware Hybrid Event-Triggered Control for Micro-Drones p.118
[16] Payload State Estimation for Cooperative Manipulation Using Multiple UAVs p.125
[17] Spiking Neural Networks for High-Speed Continuous Quadcopter Control Using Proximal Policy... . . . . p.133
[18] Design of Frontal Perching Mechanism to Vertical Surfaces for Flapping-Wing MAVs p.142
[19] Pecking Mechanism Inspired by Avian Cranial Kinesis for Flapping-Wing Aerial Manipulators p.148
[20] Flight Pattern and Experimental Flights of a Quadricopter Drone at the Edge of a Smoke Plume p.156
[21] Dielectric Barrier Discharge Plasma Actuators to Mitigate Flow Separation p.162

http://www.imavs.org/

Authors

A

Al-zubaidi, Salim p.106
Aláez, Daniel p.76
Asignacion, Abner p.69, p.125, p.142, p.148
Astrain, José Javier p.76

B

Balam-Velasco, Brianna p.25
Becerra, Pedro p.37
Briseño, Blanca p.37

C

Campoy, Pascual p.98
Castillo, Michael van Breukelen p.133
Chamorro, Francisco Anguita p.98
Chen, Ben M. p.90
Chen, Xi p.90
Chen, Yizhou p.90
Cortizas, Miguel Fernandez p.98
Costa, Laisa p.32
Croon, Guido de p.133

D

Deza, Javier Melero p.98
Durand, Sylvain p.156

F

Ferede, Robin p.133

G

Gabriel, Guilherme p.32
Gao, Zhi p.90
García-Maya, Brenda Ivette p.43
Gaskell, Maggie p.162
Gonzalez, Hernan Abaunza p.43
González-Hernández, Hugo Gustavo p.118
González-Tejeda, Fabián p.43

H

Huang, Yijun p.90
Hussain, Saad p.142, p.148

K

Kiefer, Renaud p.156
Krasinkiewicz, Robin p.162

L

Laroche, Edouard p.156
Liu, Janet p.162
Lu, Zhanghao p.51

M

Maggiani, Noah p.162
Martinez-Carranza, Jose p.18, p.25, p.82, p.112, p.118
Martinez-Torres, Cesar p.112
Martínez, Diego Alexis p.37
Matsuda, Tomoyuki p.142

O

Ohira, Yuki p.125
Onda, Yuki p.69

P

Perez-Rodriguez, Carlos p.112
Pivert, Maya p.156
Prieto, Manuel p.76

Q

Quero, Carlos Osorio p.82

R

Rodriguez, Jesus p.118
Rodríguez, Hugo p.37
Rojas, Carmen De p.98
Rojas-Perez, L. Oyuki p.25

S

Sarmento, Bruno p.32
Segui, Rafael Perez p.98
Smith, Charlie p.162
Sousa, Sacha De p.156
Stol, Karl p.106
Suzuki, Satoshi p.125, p.142, p.148

http://www.imavs.org/

Suzuki, and Satoshi p.69

T

Trinidad-Rendón, Julio Sebastián De la p.118

V

Villadangos, Jesús p.76
Vos, Reinier p.133

W

Wagter, Christophe De p.59, p.133
Wang, Jialiang p.90
Watson, Simon p.51
Weightman, Andrew p.51
Woods, Aidan p.162
Wu, Annie p.162
Wu, Qigeng p.90
Wu, Zongzhou p.90

X

Xolo-Tlapanco, Nilda p.18

Y

Yan, Haowen p.162
Yıkılmaz, Cansu p.59

Z

Zhou, Zhiyu p.90

Reviewers

Aláez, Daniel
Al-zubaidi, Salim
Anguita Chamorro, Francisco
Asignacion, Abner
Astrain, José Javier
Balam-Velasco, Brianna
Becerra, Pedro
Becker, Marcelo
Briseño, Blanca
Cabrera-Ponce, Aldrich
Campoy Cervera, Pascual
Chen, Ben M
Chen, Xi
Chen, Yizhou
Costa, Laisa
De Croon, Guido
De Rojas Pita-Romero, Carmen
De Wagter, Christophe
Ferede, Robin
Fernandez Cortizas, Miguel
Gabriel, Guilherme
Gao, Zhi
Gaskell, Maggie
Huang, Yijun
Hussain, Saad
Krasinkiewicz, Robin
Lie, Janet
Lu, Zhanghao
Maggiani, Noah
Marsicano, João Aires
Martínez, Diego Alexis
Martinez-Carranza, Jose
Matsuda, Tomoyuki
Melero Deza, Javier
Negrete-Villanueva, Marco
Ohira, Yuki
Onda, Yuki
Osorio Quero, Carlos
Perez Segui, Rafael
Prieto, Manuel
Rodríguez, Hugo
Rojas-Perez, L. Oyuki
Sarmento, Bruno
Smith, Charlie
Stol, Karl
Suzuki, Satoshi
Van Breukelen Castillo, Michael
Villadangos, Jesús
Vos, Reinier
Wang, Jialiang
Watson, Simon
Weightman, Andrew
Woods, Aidan
Wu, Annie
Wu, Qigeng
Wu, Zongzhou
Xolo-Tlapanco, Nilda
Yan, Haowen
Yıkılmaz, Cansu
Zhou, Zhiyu

http://www.imavs.org/

http://www.imavs.org/

Release

Final Release on November 14^{th} 2025.

V1.0, Rev.16819, 2026-10-17 19:18
Created by: Jose Martinez-Carranza and C. De Wagter

3D CNN and Multi-Head Attention to Relative Camera Position

Nilda G. Xolo-Tlapanco and Jose Martinez-Carranza *
Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), Puebla, México

http://www.imavs.org/

ABSTRACT

Calculating the relative position between images in Unmanned Aerial Vehicles (UAVs) is a core component in tasks such as visual odometry, SLAM, and state estimation. It enables the UAV to estimate its movement between frames using onboard sensors. In this paper, we present a spatio-temporal regression architecture combining 3D CNNs and masked attention mechanisms. The model takes a sequence of image frames and corresponding Inertial Measurement Unit (IMU) data for each frame as input and outputs the estimated relative position in meters between the images. The inclusion of IMU measurements is critical for improving robustness to motion blur, low-texture environments, and rapid maneuvers, as it provides complementary information about the UAV's linear acceleration and angular velocity. The CNN layers extract compact spatio-temporal features from the video input, while the multi-head attention layer captures temporal dependencies and contextual relations across time. The final Multi-Layer Perceptron (MLP) regresses these fused representations into a relative position estimate. We demonstrate the effectiveness of our method on the TII Drone Racing and UZH-FPV datasets.

1 INTRODUCTION

Accurate estimation of the relative camera pose between consecutive frames is a fundamental problem in computer vision and robotics, with applications in autonomous UAV navigation, visual odometry, augmented reality, and simultaneous localization and mapping (SLAM). Estimating how a camera moves through an environment using only visual data enables devices to operate in GPS-denied or dynamic environments, a crucial capability for lightweight, agile aerial robots. However, relying solely on visual input can be challenging in scenes with motion blur, low texture, or sudden accelerations [1]. To address this, inertial measurements from an Inertial Measurement Unit (IMU) provide complementary motion cues, capturing high-frequency information about the UAV's linear acceleration and angular velocity. When fused with visual data, IMU information improves pose estimation accuracy and temporal consistency, particularly in fast or ambiguous motion scenarios [2].

Traditionally, relative pose estimation has been based on geometry-based methods such as feature matching, essential matrix decomposition, and epipolar geometry [3, 4, 5]. Although effective under ideal conditions, these approaches are sensitive to image noise, motion blur, dynamic objects, and low-texture scenes (all common in UAV flight scenarios). More recently, learning-based approaches have shown promise in extracting robust representations from raw images, enabling the end-to-end estimation of camera motion without the need for explicit feature extraction or geometric modeling [6], as well as the use of only IMU information to calculate the relative pose and orientation [7, 8]. However, most existing learning methods rely on 2D convolutional networks, which fail to fully capture the temporal dynamics and motion continuity inherent in sequential image data.

To address these limitations, we propose a novel deep learning architecture that combines 3D Convolutional Neural Networks (3D CNNs) with a masked multi-head attention mechanism for robust and efficient estimation of relative camera motion with the help of inertial measurements from an IMU. The 3D CNN serves as a spatio-temporal feature extractor, capturing both appearance and motion patterns across frames, while the attention module models long-range dependencies and spatial-temporal interactions, improving the consistency and robustness of the motion estimation.

2 RELATED WORK

Traditional approaches for estimating the relative pose between two images rely heavily on geometric principles, such as epipolar geometry and Structure-from-Motion (SfM) pipelines. These methods typically involve detecting and matching feature points (like SIFT or ORB methods), estimating the fundamental or essential matrix, and decomposing it to recover relative rotation and translation (up to scale) [9]. While effective under controlled conditions, these techniques degrade significantly in low-texture regions, under motion blur, or when the scene contains dynamic elements, scenarios common in UAV operations.

With the rise of deep learning, several methods have been proposed to estimate camera motion directly from raw image data. PoseNet [10] introduced a convolutional architecture for regressing the absolute camera pose from a single image, later extended to use temporal information. DeepVO [6],

*Email address: carranza@inaoep.mx

http://www.imavs.org/

SfM-Net [11], and SelfVIO[12] focused on estimating relative motion using Recurrent Neural Networks (RNN) or convolutional architectures on image pairs or sequences. These methods bypass traditional feature extraction but often rely on 2D CNNs, which may not fully capture temporal dynamics across frames.

3D convolutional neural networks have been widely adopted in video classification [13] and action recognition [14] due to their ability to model spatio-temporal dependencies. Models like C3D [15] and I3D [16] extract features from both spatial and temporal dimensions, making them suitable for motion understanding. In the context of pose estimation, 3D CNNs offer the advantage of learning motion cues directly from short video clips. However, their use in relative pose estimation, particularly for UAVs, remains underexplored.

The introduction of Transformers [17] and their adaptation to vision tasks (ViT [18], TimeSformer [19]) have shown that self-attention can be highly effective in modeling long-range dependencies and contextual relationships in image and video data. In pose estimation, attention-based models such as ConvLSTM [20] and DROID-SLAM [21] incorporate attention layers to integrate spatial and temporal information between image sequences. These methods demonstrate the potential of attention mechanisms to enhance robustness, especially under challenging conditions.

For aerial robotics, estimating relative pose is especially critical due to limited access to GPS, high-speed maneuvers, and environmental variability. Methods like VINS-Mono [22] and ORB-SLAM3 [23] provide tightly coupled visual-inertial and SLAM-based solutions. Learning-based approaches, such of Xu et al. [24], demonstrate that compact convolutional architectures can effectively handle large disparities and motion blur in a downward-facing camera, particularly when aided by IMU signals. However, these models typically operate in a frame-to-frame setting and remain limited in capturing longer-term temporal dependencies.

Our work addresses these challenges by combining 3D CNNs for spatio-temporal motion encoding with multi-head attention mechanisms that incorporate both visual and inertial data. Specifically, we integrate IMU measurements (linear accelerations and angular velocities) as additional input streams into the attention layers, enabling the model to learn cross-modal dependencies and temporal patterns that extend beyond what vision alone can capture. This fusion improves the model's ability to disambiguate motion in scenes with textureless surfaces, motion blur, or fast UAV maneuvers. By attending jointly to visual features and inertial cues, our architecture produces more accurate and robust relative pose estimates tailored to the dynamics of agile aerial platforms.

3 METHODOLOGY

This section presents the proposed architecture for estimating the relative camera translation between consecutive frames. The network combines 3D Convolutional Neural Networks (3D CNNs) and a masked multi-head attention mechanism to model both local spatio-temporal patterns and long-range dependencies across image sequences. The output is a 3-dimensional vector corresponding to the relative displacement (x,y, and z) of the camera. Given a sequence of consecutive RGB images $I_t, I_{t+1}, \ldots, I_{t+n}$ and IMU data $IMU_t, IMU_{t+1}, \ldots, IMU_{t+n}$ representing the angular velocity $(\omega_x, \omega_y, \omega_z)$ and linear acceleration (a_x, a_y, a_z) of each image, the objective is to estimate the relative translation $t_{t \to t+1} \in R^3$ between frames I_t and I_{t+1}, assuming a known or negligible rotation. This formulation is useful in UAV applications such as visual odometry, drone racing, and inspection, where accurate estimation of egomotion is crucial. The proposed architecture consists of four main stages (Figure 1):

- **Spatio-temporal feature extraction via 3D CNN:** A stack of 3D convolutional layers processes the input sequence $X \in \mathbb{R}^{B \times C \times T \times H \times W}$, where B is the batch size, C the number of channels, T the number of frames, and $H \times W$ the image resolution (set at 122×122 in our experiments). The 3D convolutions jointly encode appearance and temporal motion patterns. After three convolutional blocks (with batch normalization and ReLU activations), the output is reshaped into a sequence of feature vectors of size 128, yielding $X_{vis} \in \mathbb{R}^{B \times L \times 128}$, where $L = C' \times H' \times W'$ is the flattened spatio-temporal dimension of the last 3D convolution, in this case different from the original dimension.

- **IMU feature encoding:** Raw IMU signals $(\omega_x, \omega_y, \omega_z, a_x, a_y, a_z)$ over a clip of T frames are stacked into a vector of dimension $6T$ (i.e., 36 when $T = 6$). A two-layer Multi-Layer Perceptron (MLP) projects this input to a 128-dimensional embedding, $X_{imu} \in \mathbb{R}^{B \times 128}$. This representation is then repeated along the sequence length L, resulting in $X_{imu} \in \mathbb{R}^{B \times L \times 128}$, to match the visual features.

- **Temporal attention via masked multi-head attention:** The visual features X_{vis} and IMU features X_{imu} are concatenated along the feature dimension, producing $X_{fused} \in \mathbb{R}^{B \times L \times 256}$. This sequence is passed to a masked multi-head self-attention module that:
  - Applies multiple attention heads to compute relationships between all pairs of feature tokens,
  - Uses an optional mask to restrict attention to valid temporal regions,
  - Enables the model to reason over global temporal dependencies and non-local motion patterns.

- **Regression head via MLP:** The attention-enhanced sequence is pooled via global average pooling over L, resulting in a fixed-size representation $\in \mathbb{R}^{B \times 256}$. A

three-layer MLP regresses the final translation vector (x, y, z).

The network is trained using supervised learning with ground truth relative translations, which are calculated with the ground truth position and orientation of the dataset.

Given two poses at timestamps t_i and t_{i+1}, each pose consists of a position vector $\mathbf{t} \in \mathbb{R}^3$ and a unit quaternion $\mathbf{q} \in \mathbb{R}^4$ representing orientation.

1. Homogeneous transformation matrix:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R}(\mathbf{q}) & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$$

2. Relative transformation: Let $\mathbf{T}_1$ and $\mathbf{T}_2$ be the transformation matrices at times t_i and t_{i+1}, respectively. The relative transformation is:

$$\mathbf{T}_{\text{rel}} = \mathbf{T}_1^{-1} \cdot \mathbf{T}_2$$

3. Output format:

$$\mathbf{T}_{\mathbf{rel}} = \begin{bmatrix} \mathbf{R}(\mathbf{q_{rel}}) & \mathbf{t_{rel}} \\ \mathbf{0} & 1 \end{bmatrix}$$

From this matrix, we extract the position and orientation to use separately.

Reconstructing global trajectory from relative poses Given a sequence of relative poses $\mathbf{t}_{\text{rel}}^i$ predicted by our network and relative orientations by the ground truth $\mathbf{q}_{\text{rel}}^i$ for $i = 0, 1, \ldots, N-1$, we reconstruct the global trajectory using the following procedure.

1. Relative transformation matrix: Each relative pose is converted to a homogeneous transformation:

$$\mathbf{T}_{\text{rel}}^i = \begin{bmatrix} \mathbf{R}(\mathbf{q}_{\text{rel}}^i) & \mathbf{t}_{\text{rel}}^i \\ \mathbf{0} & 1 \end{bmatrix} \in SE(3)$$

2. Initialization: Start at the origin with identity transformation:

$$\mathbf{T}_0 = \mathbf{I}_{4\times4}$$

3. Recursively compute global poses

$$\mathbf{T}_{i+1} = \mathbf{T}_i \cdot \mathbf{T}_{\text{rel}}^i$$

For the training, we calculate the loss with Mean Squared Error (MSE) in meters between prediction and ground truth translation t_{gt}:

$$L_{\text{pose}} = \|t_{\text{pred}} - t_{\text{gt}}\|^2 \quad (1)$$

4 EXPERIMENTS AND RESULTS

We evaluate our model on the TII Drone Racing [25] and UZH-FPV [26] datasets. The TII Drone Racing contains RGB images with illumination changes from fast (>21m/s) and aggressive quadrotor flight and the UZH-FPV dataset includes RGB images from indoor and outdoor scenes of agile flight.

The experiments were carried out on a computer equipped with an Intel Core i5-9400F, 16GB of RAM, and an NVIDIA GeForce RTX 4070 Super GPU. Due to the nature of our network, it is necessary to determine the optimal value for T (the number of frames required to subtract enough information to estimate the relative position). Therefore, we evaluate the network with T values of 2, 4, 6,8, and 10 on the UZH-FPV dataset using sequences Indoor forward facing 03, 05, and 06 for training, 07 and 09 for validation, and 10 for testing. These sequences were selected because they represent different flight trajectories and visual conditions within the same forward-facing setup. For example, training sequences (03, 05, 06) contain a mix of straight and turning maneuvers, providing sufficient variability for the model to learn motion patterns. Validation sequences (07, 09) include faster translational movements and different illumination, which are useful to monitor overfitting. Finally, sequence 10 was reserved for testing as it features a distinct trajectory not seen during training, ensuring an unbiased evaluation of generalization.

We analyze the training loss over 100 epochs. The T=10 setting is the most unstable; however, in general, the loss decreases consistently throughout training. To determine the best T value, we compare the MSE of each model for the test sequence in Table 1, with the best result obtained for T=6.

With the optimal value of T fixed at 6, we train the final model on the following UZH-FPV sequences to evaluate its effectiveness:

- Training set: Indoor forward-facing: 06, 07, 09, Outdoor forward-facing: 03, Indoor 45° downward-facing: 02, 04, 09, and Outdoor 45° downward-facing: 01
- Validation set: Outdoor forward-facing: 01 and Indoor 45° downward-facing 12.
- Testing set: Indoor forward-facing: 10, Outdoor forward-facing: 05 and Indoor 45° downward-facing: 14

With a total of 15,010 images for training, 3,253 for validation, and 3,065 for testing. In the case of the TII Drone racing dataset we use the following sequences:

- Training set: 01, 05, 08, 09, 11, and 12.
- Validation set: 02, 03, 06, and 07.
- Testing set: 04 and 10.

http://www.imavs.org/

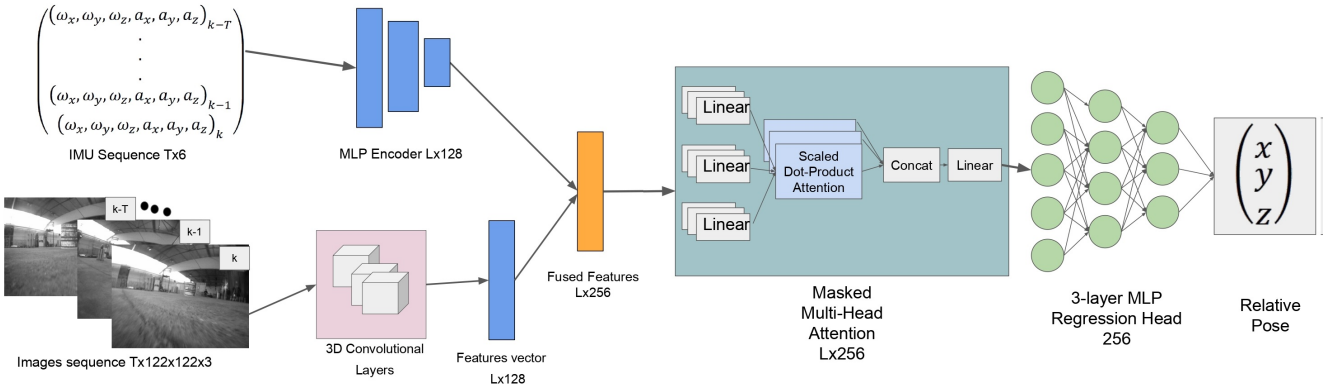


Figure 1: Overview of the proposed architecture. The model consists of four stages: (1) spatio-temporal feature extraction via a 3D CNN that encodes motion cues across frames, (2) IMU feature encoding with a lightweight MLP, (3) temporal attention using a masked multi-head attention mechanism to capture long-range dependencies, and (4) a regression head via MLP to estimate the translation vector (x, y, z), where T is the number of frames and L the new dimension of the last 3D convolution.

http://www.imavs.org/

With a total of 34,882 images for training, 23,087 for validation, and 12,008 for testing. We prepare two networks for comparison, the first and main one, 3D CNN-IMU Attention described in the methodology, and 3D CNN-Attention, where the IMU information is excluded. In the latter, the input for the multi-head attention consists only of the features provided by the 3D convolutional layers, allowing us to evaluate the importance of the IMU data.

T value	MSE
2	1.123
4	1.216
6	**0.919**
8	1.642
10	1.403

Table 1: MSE for each value of T (2,4,6,8, and 10) in testing trajectories.

We reconstruct the global position predicted by each network and present it in Figures 2 and 3, corresponding to the training and testing trajectories, respectively. Once the predictions are obtained, we compute the MSE for each trajectory and compare them with the ground truth (Table 2), where the best values are highlighted in black.

As can be observed, the network without IMU information performs better on the training trajectories—i.e., those it has already seen—compared to unseen trajectories used in validation and testing, where IMU information appears to improve performance in unfamiliar environments.

4.1 Ablation Study

To assess the importance of each component of the network, we conduct experiments with different variations of our approach:

- 3D CNN-Attention with IMU information (3D CNN-Att IMU).
- 3D CNN-Attention without IMU information (3D CNN-Att).
- 3D CNN without Attention (3D CNN without Att).
- 3D CNN with IMU information and without Attention (3D CNN IMU without Att).
- CNN with Attention without IMU (CNN- Att).

In the case of the 2D CNN baseline, to emulate the temporal encoding of the 3D CNN, we concatenate the 6 consecutive input frames into a 3×2 grid, forming a single 224× 224 image. This strategy allows the 2D CNN to process multiple frames at once, although the temporal relationships must be inferred indirectly from the spatial arrangement.

These experiments were carried out only on the UZH-FPV dataset. Training and inference times for each variation are reported in Table 3 and the MSE for each trajectory are presented in Table 4.

As observed, CNN-Attention achieves the best performance in most training cases, but not in unseen environments, where other variations of the network perform better. The main disadvantage of the CNN-based architecture compared to 3D-CNN is the significantly longer training and inference time; the 3D CNN is 96% faster in training and 54% faster in inference, which is an important factor in real-time scenarios. Training a 3D CNN can be faster because temporal information is encoded natively through 3D convolutions, whereas a 2D CNN with a frame grid wastes capacity and training time trying to infer temporal relations from a spatial layout that doesn't naturally represent time, and in the case for inference. Furthermore, the attention module plays a critical role in the

network, as removing it in "3D CNN without Att" variation leads to a noticeable drop in accuracy for most sequences, having the worst MSE scores.

Sequence	3D CNN-Att	3D CNN- IMU Att
	Train	
Indoor 06	**0.0609**	0.0836
Indoor 07	**0.0715**	0.1537
Indoor 09	**0.0162**	0.0374
Outdoor 03	**0.0883**	0.1444
Indoor 45° 2	**0.0297**	0.0420
Indoor 45° 4	**0.0227**	0.0333
Indoor 45° 9	**0.0209**	0.0388
Outdoor 45° 1	**0.0450**	0.0538
DR 01	**0.0083**	0.0114
DR 05	**0.0061**	0.0105
DR 08	**0.0057**	0.0075
DR 09	**0.0088**	0.0126
DR 11	**0.0093**	0.0128
DR 12	**0.0082**	0.0106
	Validation	
Outdoor 1	**1.3317**	1.7819
Indoor 45° 12	2.5218	**0.8214**
DR 02	0.0922	**0.0378**
DR 03	0.1029	**0.0241**
DR 06	0.2087	**0.0465**
DR 07	0.0983	**0.0218**
	Test	
Indoor 10	0.1775	**0.1711**
Outdoor 5	1.3651	**1.3574**
Indoor 45° 14	5.3050	**1.4855**
DR 04	0.2087	**0.0654**
DR 10	0.0983	**0.0347**

Table 2: MSE in meters of each sequence for our network with (3D CNN- IMU Att) and without (3D CNN-Attention) IMU information, the best values are highlighted in black.

Network	Training Time (s)	Inference Time (s)
3D CNN-Att IMU	61.60	1.481e-03
3D CNN-Att	61.06	1.332e-03
3D CNN without Att	61.24	**0.932e-03**
3D CNN IMU without Att	**59.72**	1.128e-03
CNN- Att	1532.51	2.895e-03

Table 3: Mean training time for 100 epochs and inference time for the testing sequences in seconds. The best values are highlighted in bold.

5 CONCLUSION

We presented a novel deep learning architecture for estimating relative camera pose, a critical task in computer vision and robotics, particularly in Unmanned Aerial Vehicles (UAVs), where onboard computational resources are often limited. Our method combines 3D Convolutional Neural Networks (3D CNNs) with a masked multi-head attention mechanism that fuses visual and inertial data, specifically inputs from a monocular camera and an Inertial Measurement Unit (IMU), both commonly available on UAV platforms. Through extensive experiments, we evaluated the effectiveness of our architecture under different configurations and demonstrated that incorporating inertial measurements improves generalization to unseen environments, reducing the MSE error in all test trajectories. Additionally, the use of 3D CNNs contributes to 96% faster in training and 54% faster in inference and more efficient inference by leveraging spatio-temporal motion cues.

For future work, we plan to evaluate the robustness of our method in real-world flight scenarios perform broader comparisons with state-of-the-art visual–inertial odometry and SLAM approaches, and extend the model to estimate full six degrees of freedom poses (translation and rotation) using additional regression heads.

ACKNOWLEDGEMENTS

The author Nilda G. Xolo-Tlapanco, thanks SECIHTI in Mexico for their scholarship with CVU number 1230561.

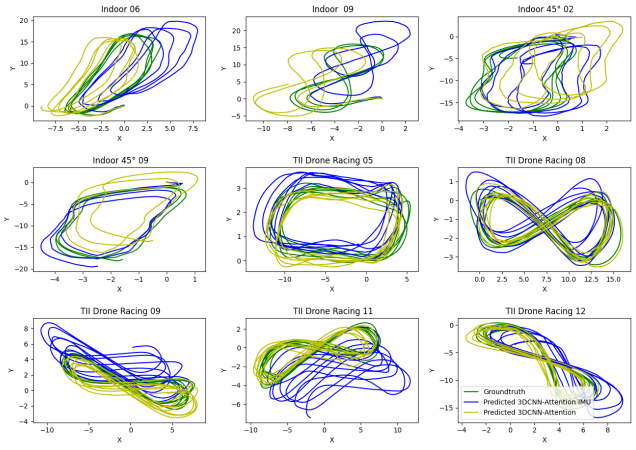


Figure 2: Reconstruction of some trajectories used for training, ground truth (green), predicted by 3D CNN-Attention IMU (blue), and predicted by 3D CNN-Attention (yellow).

REFERENCES

[1] Licong Zhuang, Xiaorong Zhong, Linjie Xu, Chunbao Tian, and Wenshuai Yu. Visual slam for unmanned aerial vehicles: Localization and perception. *Sensors*, 24(10), 2024.

http://www.imavs.org/

Sequence	3D CNN-Att	3D CNN-Att IMU	3D CNN without Att	3D CNN IMU without Att	CNN- Att
			Train		
Indoor 06	0.0609	0.0836	1.3746	0.2071	**0.0283**
Indoor 07	0.0715	0.1537	1.5965	0.3428	**0.0597**
Indoor 09	0.0162	0.0374	0.3553	0.1351	**0.0134**
Outdoor 03	0.0883	0.1444	1.4994	0.3608	**0.0879**
Indoor 45° 2	**0.0297**	0.04204	0.3966	0.0764	0.0314
Indoor 45° 4	0.0227	0.0333	0.2639	0.0604	**0.0210**
Indoor 45° 9	**0.0209**	0.0388	0.4918	0.0528	0.0433
Outdoor 45° 1	0.0450	0.0538	0.3537	0.0662	**0.0305**
			Validation		
Outdoor 1	**1.3317**	1.7819	1.8531	0.0662	1.6686
Indoor 45° 12	2.5218	0.8214	0.8060	**0.0662**	1.2969
			Test		
Indoor 10	0.1775	**0.1711**	0.5431	0.3294	0.2086
Outdoor 5	1.36514	1.3574	1.3030	1.3754	**1.3313**
Indoor 45° 14	5.3050	1.4855	2.5718	**1.2046**	2.7904

Table 4: MSE in meters of each sequence for different variations of our network, the best values are highlighted in black.

http://www.imavs.org/

[2] Davide Scaramuzza and Zichao Zhang. *Visual-Inertial Odometry of Aerial Robots*, page 1–9. Springer Berlin Heidelberg, 2020.

[3] Hugh Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

[4] Tserennadmid Tumurbaatar and Taejung Kim. Comparative study of relative-pose estimations from a monocular image sequence in computer vision and photogrammetry. *Sensors*, 19(8), 2019.

[5] D. Nister. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, 2004.

[6] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2050, 2017.

[7] Changhao Chen, Xiaoxuan Lu, Andrew Markham, and Niki Trigoni. Ionet: learning to cure the curse of drift in inertial odometry. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press, 2018.

[8] Changhao Chen, Chris Xiaoxuan Lu, Johan Wahlström, Andrew Markham, and Niki Trigoni. Deep neural network based inertial odometry using low-cost inertial measurement units. *IEEE Transactions on Mobile Computing*, 20(4):1351–1364, 2021.

[9] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, March 2004.

[10] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2938–2946, 2015.

[11] Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, and Katerina Fragkiadaki. Sfm-net: Learning of structure and motion from video, 2017.

[12] Yasin Almalioglu, Mehmet Turan, Muhamad Risqi U. Saputra, Pedro P.B. de Gusmão, Andrew Markham, and Niki Trigoni. Selfvio: Self-supervised deep monocular visual–inertial odometry and depth estimation. *Neural Networks*, 150:119–136, 2022.

[13] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[14] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, 2013.

[15] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal

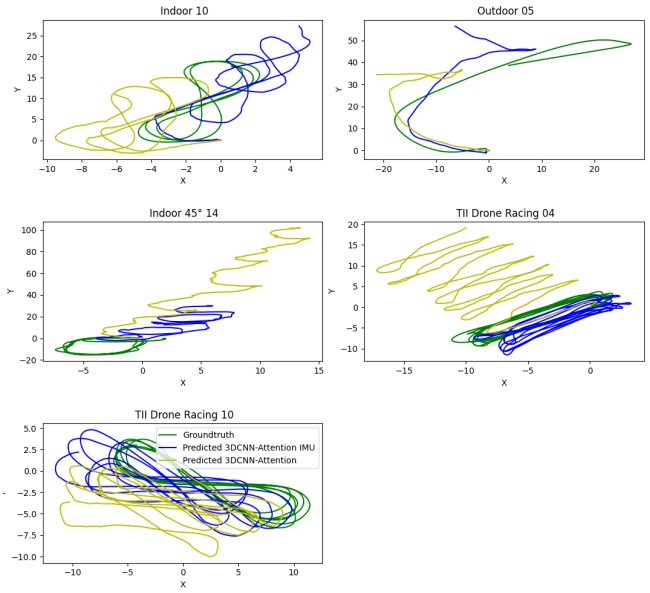


Figure 3: Reconstruction of the trajectories used for testing, ground truth (green), predicted by 3D CNN-Attention IMU (blue), and predicted by 3D CNN-Attention (yellow).

features with 3d convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015.

[16] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[18] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[19] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning (ICML)*, July 2021.

[20] Sangni Xu, Hao Xiong, Qiuxia Wu, and Zhiyong Wang. Attention-based long-term modeling for deep visual odometry. In *2021 Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8, 2021.

[21] Zachary Teed and Jia Deng. DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras. *Advances in neural information processing systems*, 2021.

[22] 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.

[23] Carlos Campos, Richard Elvira, Juan J. Gomez, José M. M. Montiel, and Juan D. Tardós. ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.

[24] Yingfu Xu and Guido C. H. E. de Croon. Cnn-based ego-motion estimation for fast mav maneuvers. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7606–7612, 2021.

[25] Michael Bosello, Davide Aguiari, Yvo Keuter, Enrico Pallotta, Sara Kiade, Gyordan Caminati, Flavio Pinzarrone, Junaid Halepota, Jacopo Panerati, and Giovanni Pau. Race against the machine: A fully-annotated, open-design dataset of autonomous and piloted high-speed flight. *IEEE Robotics and Automation Letters*, 9(4):3799–3806, 2024.

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

http://www.imavs.org/

A Review of Visual SLAM Systems and Single-Image Depth Estimation for Challenging Scenarios

Brianna A. Balam-Velasco, L. Oyuki Rojas-Perez, and Jose Martinez-Carranza *
Computer Science Department, Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), Puebla 72840, Mexico.

http://www.imavs.org/

Abstract

This article reviews monocular SLAM systems operating in challenging scenarios, with a particular emphasis on agricultural environments. Key metrics such as pose accuracy, processing time, and trajectory fidelity are examined across complex datasets. To address the known limitations of monocular systems in textureless or repetitive scenes, we explore the integration of single-image depth estimation models, including Depth Anything and ZoeDepth. We evaluate a hybrid SLAM pipeline based on ORB-SLAM2 enhanced with synthetic depth maps and dynamic keypoint adaptation, tested on the Rosario Dataset. The findings demonstrate improvements in localisation accuracy, robustness, and global trajectory consistency under real-world agricultural conditions, establishing a foundation for future research in SLAM enhancement via monocular depth priors.

1 Introduction

Visual SLAM (Simultaneous Localisation and Mapping) plays an important role in enabling autonomous navigation in environments where GPS is unreliable or unavailable. Among the many SLAM modalities, monocular SLAM systems have gained attention due to their minimal hardware requirements and potential for deployment on lightweight platforms such as drones. However, traditional monocular SLAM faces persistent challenges in environments with low texture, repetitive structures, or dynamic lighting conditions-scenarios common in real-world agricultural and extraterrestrial settings.

To address these challenges, recent research has explored augmenting monocular systems with single-image depth estimation models. These approaches aim to bridge the gap between RGB-only and RGB-D SLAM by generating synthetic depth information that can improve tracking robustness and reduce scale drift. This paper presents a comprehensive review of monocular SLAM systems under difficult environmental conditions and explores how the integration of learning-based depth estimation methods, such as Depth Anything, ZoeDepth, and MiDaS, can enhance system performance.

*Corresponding author: carranza@inaoep.mx

By using ORB-SLAM2 with real-time synthetic depth maps and adaptive feature modulation, we evaluate the proposed hybrid strategy on the Rosario Dataset [1], a benchmark for structured and unstructured fields, and across the six sequences, we report qualitative trajectory alignments and quantitative metrics, highlighting the feasibility and limits of image-based depth priors for strengthening monocular SLAM where conventional methods often fail.

2 Related Work

The development of visual SLAM has been fundamental for autonomous navigation in GPS-denied environments. Early monocular approaches, such as PTAM [2] and MonoSLAM [3], enabled real-time performance with a single camera but were limited by scale ambiguity and poor performance in low-texture scenes. ORB-SLAM and ORB-SLAM2 [4] addressed these issues by incorporating loop closure, map reuse, and relocalisation. These systems later evolved into visual-inertial frameworks like OpenVINS [5], which improved robustness in dynamic and GPS-denied scenarios [6]. More recent efforts have integrated monocular and multi-view depth estimation to improve scale recovery, as demonstrated by Zhang et al. [7] with a tightly coupled SLAM system for indoor environments.

In parallel, semantic information has been explored to enhance localisation in visually ambiguous scenes. For instance, Martinez-Carranza et al.[8] combined natural language with YOLOv8 to support metric SLAM. Similarly, UAV-based studies have applied CNN-based RGB-D perception for GPS-denied localisation[9], and monocular cues for landing detection [10] and obstacle avoidance [11]. These contributions illustrate how combining geometric and semantic data can increase robustness in complex environments.

To further address challenges like visual degradation and dynamic motion, hybrid SLAM frameworks have emerged. Gallagher et al.[12] proposed a sparse-dense system that merges traditional feature tracking with dense depth prediction for increased resilience outdoors. Scherer[13] developed a GPS-free solution for MAVs using RGB-D constraints in low-altitude flight. However, RGB-D sensors remain limited by short range, lighting sensitivity, and vulnerability to fast motion. To overcome these constraints, Ghosh and Gallego [14] introduced a stereo depth method for event cameras, suitable for high-speed or low-light conditions. Gao et al. [15] provide a comprehensive review of hybrid stereo systems, highlighting the benefits of combining geometry with

http://www.imavs.org/

learned depth.

As an alternative to more complex sensor setups, monocular depth estimation has proven effective for lightweight robotic platforms. Pellegrin et al.[16] proposed a single-image model tailored to aerial views, while Alquisiris et al.[17] employed optical flow for real-time depth on mobile robots. Rojas-Perez et al. [18] introduced a decentralised monocular SLAM strategy for coordinating UAVs in GPS-denied environments using learned priors. These works highlight the adaptability of monocular estimation in constrained systems.

Recent advances have focused on transformer-based and zero-shot models to improve generalisation in depth prediction. AdaBins [19] introduced adaptive binning for depth regression, ZoeDepth [20] combined metric and relative depth for zero-shot transfer, and Depth Anything [21] leveraged large-scale pretraining to estimate depth in novel environments. Integrated into SLAM pipelines, these models help reduce scale drift and mapping errors. Additionally, synthetic depth priors have proven useful in low-texture or poorly lit conditions [22]. Bladh [23] evaluated AdaBins, MiDaS, and Monodepth2 within ORB-SLAM3, reporting improved rotational consistency and suggesting that synthetic depth can substitute conventional sensors in some cases.

Despite these advances, monocular SLAM still faces challenges in repetitive or texture-sparse environments. In this paper, we propose a hybrid approach that combines monocular depth priors with adaptive feature modulation to simulate RGB-D input from a single RGB stream. This system is evaluated in real-world agricultural environments, where traditional SLAM pipelines often fail due to texture scarcity and scene variability.

3 METHODOLOGY

We propose a theoretical strategy to improve the localisation performance of ORB-SLAM2 in unstructured environments by generating synthetic depth images from monocular images and using them to simulate RGB-D operation. The aim is to overcome the scale ambiguity and reduce tracking robustness typically associated with monocular SLAM, particularly in environments characterised by low texture, inconsistent illumination, or repetitive structures.

The proposed approach comprises two main conceptual contributions; first, we introduce a method to infer and adapt scene depth using spatial statistics, enabling monocular SLAM to approximate the metric scale, and we propose a dynamic visual feature modulation mechanism that adjusts the number of extracted keypoints based on interframe motion, improving tracking robustness under varying conditions.

3.1 Scale recovery through depth compensation

Monocular SLAM lacks inherent metric information; to overcome this, we propose a depth compensation method that generates synthetic metric depth maps by fusing predictions from *Depth Anything* and *ZoeDepth* estimators.

Our method employs a hybrid strategy, where *Depth Anything* generates dense, relative depth maps from each RGB frame, and while these maps preserve structural consistency, they lack a fixed metric scale. To address this, we periodically apply *ZoeDepth* to a cropped, structurally grounded region of the image to obtain a metric reference, which then allows us to compute a scaling factor by comparing the local depth variation from *Depth Anything* with the global depth range from *ZoeDepth*.

To ensure temporal coherence between consecutive frames, we apply a smoothing function to the maximum depth estimation using an exponentially weighted moving average:

$$\text{depth\_max}_t = \alpha \cdot \text{depth\_max}_{t-1} + (1 - \alpha) \cdot r_{\text{avg}} \quad (1)$$

where α is a smoothing coefficient and r_{avg} is the average of recent scaling factors. This formula maintains a stable and consistent depth representation over time, supporting pose estimation and trajectory reconstruction in the simulated RGB-D configuration of ORB-SLAM2.

And, to enable the integration of synthetic depth images into the RGB-D pipeline of ORB-SLAM2 using monocular input, it was defined a scaling strategy that transforms relative depth predictions into scale-consistent metric maps. We first periodically apply *ZoeDepth* to a cropped image region to compute the global depth range:

$$dr = \max(\text{depth}) - \min(\text{depth}) \quad (2)$$

This value serves as a reference for calibrating the relative predictions. Next, for every frame, we use *Depth Anything* to generate a dense, relative depth map; this map is then normalized, inverted, and scaled using the global range dr, for each pixel p in the region of interest, we calculate a local scaling factor:

$$r = \frac{(p - \min)}{ds} \cdot dr \quad (3)$$

where $ds = \max - \min$ is the local depth variation.

Finally, it is rescaled the entire depth map using the smoothed maximum depth value:

$$\text{depth}_{\text{metric}} = \left(1.0 - \frac{\text{depth}_{\text{norm}}}{255.0}\right) \cdot \text{depth\_max} \quad (4)$$

This produces a synthetic depth image that is both dense and scale-consistent, enabling monocular input to be treated as if it were obtained from a depth sensor.

3.2 Adaptive feature modulation based on motion analysis

In addition to spatial enhancement, a dynamic visual tracking mechanism is proposed that adjusts the number of keypoints extracted per frame. Traditional SLAM systems typically employ a fixed feature count, which may lead to inefficiencies or instability when operating in dynamic or visually complex environments.

To address this, a motion-aware adjustment strategy is defined that monitors translation and rotation between consecutive camera poses:

$$\Delta t = \|t_1 - t_0\|, \quad \Delta R = \|R_1 - R_0\| \tag{5}$$

where t_0, t_1 represent positions and R_0, R_1 are the corresponding orientations. In cases of significant motion or sudden changes in direction, the number of features is increased to preserve tracking accuracy; in contrast, during slow or stable movements, the feature count is reduced to improve computational efficiency.

This modulation is conducted incrementally and constrained by predefined thresholds to ensure smooth transitions and avoid performance oscillations.

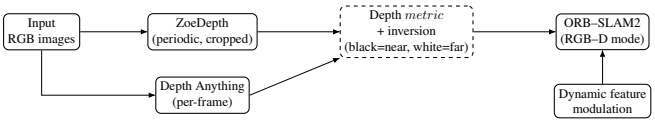


Figure 1: Proposed system pipeline, ZoeDepth periodically estimates dr on cropped frames; Depth Anything provides dense depth. Scaled, inverted maps (black=near, white=far) feed ORB–SLAM2 (RGB–D) with dynamic feature modulation.

4 EXPERIMENTS

By using sequences from the Rosario Dataset, we evaluate the system's performance under realistic agricultural conditions, focusing on how depth augmentation affects trajectory accuracy and robustness; each experiment is analyzed both visually and through quantitative error metrics to assess tracking consistency and global alignment.

4.1 Rosario Dataset

The Rosario Dataset [1] is specifically designed to assess visual localisation and mapping algorithms in realistic agricultural settings; it was captured under natural outdoor conditions, offering challenging scenarios typical of agricultural fields, including repetitive visual patterns, low-textured surfaces, dynamic illumination, and environmental variability.

Comprised of six distinct sequences, this dataset provides synchronized multi-sensor data for evaluating visual and visual-inertial SLAM systems. Key characteristics include high-resolution stereo imagery at 672×376 pixels and 15 Hz, 6-axis IMU data at 140 Hz, and a ground truth reference from wheel odometry and precise GPS-RTK positional information at 5 Hz.

These characteristics make the Rosario Dataset especially suitable for testing SLAM performance in challenging, real-world scenarios, allowing for a comprehensive benchmarking of the robustness, accuracy, and adaptability of monocular SLAM and depth estimation techniques.

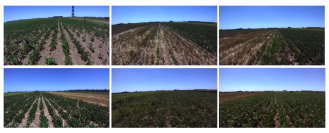


Figure 2: The Rosario dataset samples.

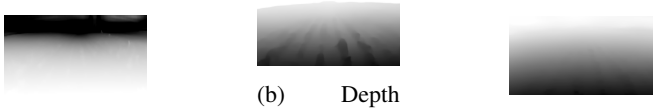
4.2 Monocular depth estimation

Depth estimation from single images has evolved rapidly, transitioning from classical convolutional neural networks to transformer-based and zero-shot learning. Early efforts, such as Eigen et al.'s multi-scale CNN [24], struggled to generalise and lacked metric accuracy, while later approaches like AdaBins [19] introduced improvements in both generalisation and precision by using an adaptive binning strategy.

ZoeDepth [20] represents a recent advance in zero-shot depth estimation; by combining relative and metric training signals, it can predict metric depths without requiring fine-tuning on each new domain. The model performs particularly well in out-of-distribution settings, but due to its high computational cost, it was used to generate global metric depth references at a lower frequency rather than for real-time processing, so depth map inversion was not implemented.

In contrast, Depth Anything [21] adopts a vision-language pretraining strategy and is trained on over 62 million images; its lightweight nature allowed for real-time synchronisation. The model produces inherently relative, highly generalisable, and visually consistent dense depth maps, so the inversion of depth values was applied, significantly facilitating clearer visual interpretation and integration with the RGB-D pipeline of ORB-SLAM2.

MiDaS [25], another monocular depth estimation model, leverages vision transformer architectures to enhance generalisation. The Swin2-Tiny variant provided a good balance of computational efficiency and depth estimation quality, achieving stable real-time synchronisation; thus, depth inversion was also applied; however, MiDaS produced slightly blurrier textures compared to Depth Anything, impacting feature matching quality and localisation accuracy.



(a) ZoeDepth (b) Depth Anything (c) MiDaS

Figure 3: Comparison of depth maps generated from the three models (Sequence 01). (a) ZoeDepth (original convention), (b) Depth Anything (inverted convention), (c) MiDaS (Swin2-Tiny, inverted convention).

Depth map inversion was strategically implemented for Depth Anything and MiDaS to facilitate visual interpretation and streamline integration with ORB-SLAM2's RGB-D

http://www.imavs.org/

pipeline; conversely, ZoeDepth was used without inversion due to computational limitations, serving as a periodic global depth reference.

4.3 Rosario Dataset Experiments

The effectiveness of the proposed methodology was evaluated using sequences from the Rosario Dataset [1]; the ORB-SLAM2 system was operated in RGB-D mode, with monocular RGB images augmented in real-time by synthetic depth maps from the Depth Anything model. These maps were produced at 15Hz, matching the frame rate of the `rosbag` stream to ensure temporal alignment.

The first sequence involved a semi-structured agricultural path with long linear stretches and sparse visual variation. The estimated trajectory (Fig. 4) successfully maintained global consistency along the main linear portions, but noticeable deviations appeared at the initial segment and sharp turns; vertical consistency was well preserved.

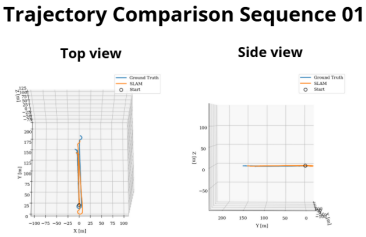


Figure 4: Comparison of ground truth and ORB-SLAM2 trajectory for Sequence 01.

Complementary error analysis (Fig. 5) indicates that the Absolute Position Error (APE) increases at the tight turns and endpoints, but drift is kept low on long linear extensions.

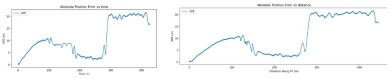


Figure 5: Sequence 01 absolute position error (APE) time (left) and distance (right).

Sequence 02 introduced increased variability in vegetation and texture, which initially enhanced SLAM tracking, but repetitive patterns later reappeared, challenging continuous feature association. The estimated trajectory (Fig. 6) successfully reproduced the global structure, including extended straight sections, but discrepancies became more pronounced at turning points and curved portions; the side view, however, showed strong consistency.

Complementary error analysis (Fig. 7) shows that the APE increases almost monotonically, with larger increments around turns, indicating cumulative drift along the long straight sections.

Sequence 03 exhibited linear motion with denser and more structured weed patterns, providing robust and stable



Figure 6: Comparison of ground truth and ORB-SLAM2 trajectory for Sequence 02.

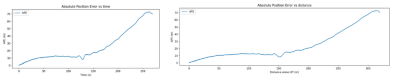


Figure 7: Sequence 02 absolute position error (APE) time (left) and distance (right).

visual features that significantly enhanced tracking. The SLAM-estimated path (Fig. 8) aligns closely with the ground truth along most linear sections, particularly in the early part of the sequence; however, noticeable deviations arise in the latter half, especially around the long curved descent and at the endpoints. The side view reveals a strong vertical alignment between both trajectories.



Figure 8: Comparison of ground truth and ORB-SLAM2 trajectory for Sequence 03.

As Figure 9 shows, the APE remains low and stable initially but increases in stages with local peaks appearing around the curved segment.

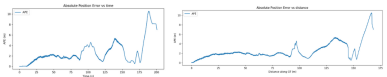


Figure 9: Sequence 03 absolute position error (APE) time (left) and distance (right).

Sequence 04 presented an environment similar to Sequence 03, characterized by repetitive and uniform textures; despite these challenges, monocular depth augmentation allowed ORB-SLAM2 to maintain trajectory estimation throughout the sequence. The estimated trajectory (Fig. 10) shows an almost perfect alignment with the ground truth, with

http://www.imavs.org/

a near overlap of both paths along the long straight segment and no noticeable vertical drift.

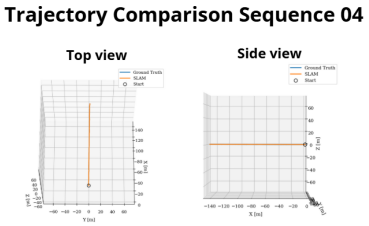


Figure 10: Comparison of ground truth and ORB-SLAM2 trajectory for Sequence 04.

Error analysis (Fig. 11) shows that the APE remains at the centimeter level throughout the entire run, indicating insignificant drift.

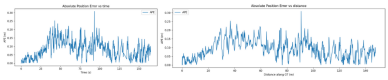


Figure 11: Sequence 04 absolute position error (APE) time (left) and distance (right).

Sequence 05 presented substantial difficulties due to an increased presence of repetitive crop patterns interspersed with abrupt changes in weed texture, which significantly challenged depth estimation and keypoint matching reliability. The estimated trajectory (Fig. 12) initially aligns with the ground truth but rapidly diverges, with significant drift evident in both top and side views, particularly during transitions between repetitive rows and abrupt texture changes.



Figure 12: Comparison of ground truth and ORB-SLAM2 trajectories for Sequence 05.

Figure 13 shows that the APE grows steadily, peaking around the middle of the run and remaining large at all times, coinciding with strong flat drift and noticeable vertical bias.

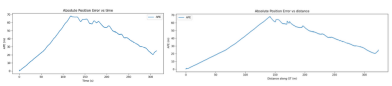


Figure 13: Sequence 05 absolute position error (APE) time (left) and distance (right).

Sequence 06 represents one of the most challenging trials, featuring long, repetitive crop rows with minimal textural distinction, which significantly hinders both keypoint extraction and depth inference. The ORB-SLAM2 system was unable to maintain trajectory accuracy; the estimated trajectory (Fig. 14) fails to capture the loop structure and deviates progressively from the ground truth, while the side view reveals a consistent vertical alignment.

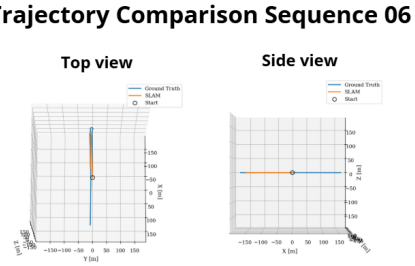


Figure 14: Comparison of ground truth and ORB-SLAM2 trajectory for Sequence 06.

Complementary error analysis (Fig. 15) shows that the APE increases sharply in the final stretch, with the distance view following an almost linear trend on long straights and steepening towards the end.

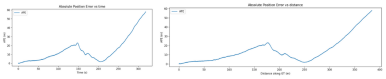


Figure 15: Sequence 06 absolute position error (APE) time (left) and distance (right).

4.4 Discussion

The experiments performed on the Rosario Dataset reveal valuable insights into integrating monocular depth estimation into a visual SLAM pipeline under challenging agricultural conditions. Qualitatively, the proposed hybrid approach, which uses Depth Anything for real-time relative estimation and ZoeDepth for periodic metric correction, enables the system to better preserve spatial coherence and trajectory fidelity than traditional monocular SLAM. The quantitative results in Table 1 confirm that while residual drift is still present, the system maintains consistent alignment in structured segments; this improvement stems from the adaptive modulation of feature extraction and dynamic scaling of depth maps.

Sequences 04 (RMSE of 0.05 m) and 03 (RMSE of 3.20 m) achieved the lowest errors, reflecting the system's ability to track accurately in stable, well-structured environments, while Sequence 01 (RMSE of 7.33 m) showed moderate accuracy. In contrast, sequences 02, 05, and 06 exhibited substantially higher errors due to texture-scarce or highly repetitive segments and re-localization challenges. Despite these errors, the overall shape and orientation of the trajec-

http://www.imavs.org/

tories were generally maintained, demonstrating the system's ability to preserve route consistency even when local accuracy was degraded.

Table 1: Trajectory error metrics on the Rosario Dataset using the proposed monocular depth-augmentation approach.

Seq.	RMSE (m)	MAE (m)	Median AE (m)	Max (m)	Min (m)
01	7.33	6.55	7.48	12.14	1.08
02	32.87	25.36	14.35	72.89	0.00
03	3.20	2.31	1.42	9.27	0.05
04	0.05	0.04	0.03	0.22	0.00
05	45.52	41.32	44.90	68.11	0.00
06	19.82	15.53	12.05	54.81	0.09
Avg.	**18.13**	**15.19**	**13.37**	**36.24**	**0.21**

Purely monocular ORB-SLAM2 did not initialize in any of the sequences, highlighting the difficulty of repetitive row crop patterns. In contrast, the proposed hybrid approach initialized in all sequences, demonstrating substantially greater operational robustness; this indicates that periodic metric cues and dense per-frame relative depth are crucial for consistent tracking in challenging scenarios.

While the proposed system improves localisation, practical trade-offs remain. **ZoeDepth** is applied selectively on cropped regions due to its high computational cost, limiting the frequency of global depth corrections. In contrast, **Depth Anything** enables real-time synchronisation and produces dense relative depth maps; however, its non-metric nature can cause local inconsistencies in visually repetitive or texture-sparse regions.

In summary, the combination of **Depth Anything** for dense real-time inference and **ZoeDepth** for metric scaling allows the pipeline to achieve a consistent, globally coherent trajectory; while residual drift remains a limitation, the approach significantly improves localization robustness over traditional monocular SLAM, which often fails in these environments.

5 CONCLUSIONS

A hybrid monocular SLAM framework is presented, which integrates synthetic depth estimation from *Depth Anything* and *ZoeDepth* with adaptive keypoint modulation, enhancing localization robustness in visually complex and repetitive environments. Using real-time relative depth inference and periodic metric corrections, the system enables ORB-SLAM2 to approximate RGB-D performance with monocular input.

Evaluations on the Rosario Dataset demonstrated consistent improvements in global trajectory alignment and tracking stability, particularly in sequences with structured textures and repeatable features; although residual drift remained a challenge, the system preserved the overall trajectory configuration. Quantitative metrics, including RMSE and absolute trajectory errors, corroborated these findings, highlighting the practical value of synthetic depth priors in extending monocular SLAM to environments where traditional pipelines often fail.

Future research will focus on incorporating loop closure based on learned descriptors, using lightweight and uncertainty-sensitive depth models, and optimizing the system for embedded applications. Other interesting directions include long-term pipeline operation and scalable mapping in dynamic, unstructured environments; the contribution lays a more solid foundation for applying monocular SLAM in practical scenarios such as field robots, planetary exploration, and agriculture.

REFERENCES

[1] Taihú Pire, Juan Corti, Guillermo L. Grinblat, Diego Di Lernia, Sebastián Bianchi, and Pablo De Cristóforis. The rosario dataset: Multisensor data for localization and mapping in agricultural environments. In *Proceedings of the XX Congreso Argentino de Ciencias de la Computación (CACIC)*, pages 1–10. RedUNCI, 2019.

[2] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234. IEEE, 2007.

[3] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[4] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[5] Patrick Geneva, Kyle Eckenhoff, Yulin Lee, and Guoquan Huang. Openvins: A research platform for visual-inertial estimation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4666–4672. IEEE, 2020.

[6] Michael Burri, Jakob Nikolic, Philip Gohl, Thomas Schneider, Johannes Rehder, Sami Omari, Markus Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016.

[7] X. Wang, Z. Zhang, and L. Li. A tightly-coupled dense monocular visual-inertial odometry system with lightweight depth estimation network. *Applied Soft Computing*, 171:112809, 2025.

[8] Jose Martinez-Carranza, David I. Hernandez-Farias, Leticia Oyuki Rojas-Perez, and A. A. Cabrera-Ponce. Language meets yolov8 for metric monocular slam. *Journal of Real-Time Image Processing*, 20(59), 2023.

http://www.imavs.org/

[9] Jose Martinez-Carranza, Leticia Oyuki Rojas-Perez, A. A. Cabrera-Ponce, and R. Munguia-Silva. Combining deep learning and rgbd slam for monocular indoor autonomous flight. In *15th Mexican International Conference on Artificial Intelligence (MICAI)*, Guadalajara, Jalisco, Mexico, 2018.

[10] Leticia Oyuki Rojas-Perez, R. Munguia-Silva, and Jose Martinez-Carranza. Real-time landing zone detection for uavs using single aerial images. In *10th International Micro Air Vehicle Conference (IMAV)*, Melbourne, Australia, 2018.

[11] Leticia Oyuki Rojas-Perez and Jose Martinez-Carranza. Metric monocular slam and colour segmentation for multiple obstacle avoidance in autonomous flight. In *IEEE 4th Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, Linköping, Sweden, 2017.

[12] Liam Gallagher, Vageesh Kumar, Senthil Yogamani, and John McDonald. A hybrid sparse-dense monocular slam system for autonomous driving. In *2021 European Conference on Mobile Robots (ECMR)*, pages 1–8. IEEE, 2021.

[13] Sebastian A. Scherer. *Efficient Visual-Inertial SLAM for Micro Aerial Vehicles*. PhD thesis, University of Tübingen, Germany, 2016.

[14] Arnab Ghosh and Guillermo Gallego. Multi-event-camera depth estimation and outlier rejection by refocused events fusion. *arXiv preprint arXiv:2207.10494*, 2022.

[15] Boyu Gao, Haoxiang Lang, and Jing Ren. Stereo visual slam for autonomous vehicles: A review. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1840–1847, Toronto, Canada, 2020. IEEE.

[16] Luca Pellegrin and Jose Martinez-Carranza. Towards depth estimation in a single aerial image. *International Journal of Remote Sensing*, 41(1):1–19, 2020.

[17] O. Alquisiris-Quecha and Jose Martinez-Carranza. Depth estimation using optical flow and cnn for the nao robot. In *18th Mexican International Conference on Artificial Intelligence (MICAI)*, Xalapa, Veracruz, Mexico, 2019.

[18] Leticia Oyuki Rojas-Perez and Jose Martinez-Carranza. Flight coordination of mavs in gps-denied environments using a metric visual slam. In *11th International Micro Air Vehicle Conference (IMAV)*, Madrid, Spain, 2019.

[19] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. *arXiv preprint arXiv:2011.14141*, 2020.

[20] S. F. Bhat, R. Birkl, D. Wofk, P. Wonka, and M. Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth. 2023.

[21] Zhen Li et al. Depth anything: Unleashing the power of large-scale unlabelled data. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2023.

[22] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose Neira, Ian Reid, and John J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[23] Daniel Bladh. Deep learning-based depth estimation models with monocular slam: Impacts of pure rotational movements on scale drift and robustness. Master's thesis, Linköping University, 2023.

[24] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *arXiv preprint arXiv:1406.2283*, 2014.

[25] Rene Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *arXiv preprint arXiv:1907.01341*, 2019.

http://www.imavs.org/

Synthetic Dataset Generation for Object Detection in MAV Applications

Guilherme Santos Gabriel, Bruno Andreoni Sarmento*, Laisa Costa De Biase and Marcelo Knorich Zuffo
University of São Paulo, R. da Reitoria, 374, São Paulo

http://www.imavs.org/

Abstract

The growing demand for accurate object detection across diverse applications has intensified the need for efficient and adaptable dataset generation techniques. In this study, we explore the use of synthetic datasets in the domain of Micro Air Vehicles (MAVs), addressing both indoor and outdoor detection tasks. We propose two procedural geometry-based approaches using Blender, a 3D modeling software, to simulate datasets for drone-based landing pad detection and animal detection in competition scenarios. These datasets were validated through participation in the Brazilian Robotics Competition 2023 and the IMAV 2024. Our results demonstrate that synthetic datasets can significantly reduce annotation effort, offer high adaptability, and maintain dataset quality, provided qualified personnel are available. We compare these synthetic approaches with traditional real-world dataset generation methods and highlight their potential to streamline dataset creation in robotics and AI.

1 Introduction

With the growing adoption of object detection technologies across various fields, the need for high-quality datasets has become increasingly critical. In particular, applications involving *Micro Aerial Vehicles* (MAVs), commonly known as drones, require task-specific datasets due to variations in operational scenarios and detection distances, thereby intensifying dataset development challenges.

The traditional process of building datasets consist in capturing real world images and manually annotating then, however it is too resource-intensive and time-consuming. This process becomes unfeasible for highly specialized situations (as with the MAV domain), thus some new processes seeks to bypass the real world interaction, with varying degrees of success.

We briefly present these different techniques and investigate the application of synthetic datasets created by 3D modeling and rendering. This investigation led to the development of two datasets validated in MAV competitions and also highlighted some strengths and weaknesses of the methods used.

*Email addresses: guilhermesangabriel@usp.br, bruno_as@usp.br

2 State of the Art

Recent research has explored the use of synthetic datasets as an alternative to traditional real-world data collection. One notable example is the *FireFly* dataset [1], developed with Unreal Engine 4 to simulate a forest environment with realistic fire-induced embers. Designed for firefighting-drone scenarios, this dataset demonstrated that synthetic data can achieve statistically comparable performance to real-world datasets, thereby validating its use in practical applications.

Synthetic datasets for object detection are typically generated using two main approaches: *image-based* and *procedural geometry-based* methods [2], as presented in Figure 1.

The image-based approach relies on existing images to generate new ones, often using generative techniques such as Transformer models, diffusion models, and autoencoders. While effective, these methods requires a realistic image dataset beforehand.

In contrast, the procedural-geometry approach uses 3-D modeling tools to create fully controllable and realistic scenes, offering greater flexibility in modifying elements such as lighting, backgrounds, camera positions, and object placements.

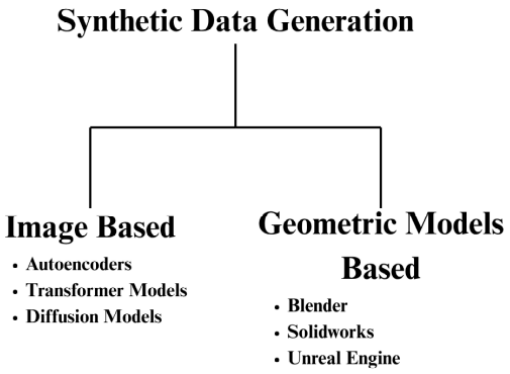


Figure 1: Taxonomy showing some options for synthetic data creation.

Game engines and 3D CAD softwares are required tools for the synthetic data generation:

- **Game engines** — Provide tools for procedural animation and scene variation. For instance, the *PeopleSansPeople* dataset generator [3] built on Unity enables semi-automated creation of labeled datasets for person detection and human-pose estimation.

- **3-D CAD tools** — Offer object modeling and realistic rendering capabilities. We could cite the software: Blender, Maya, and SolidWorks Visualize [4].

Although these tools (Figure 1) are not traditionally intended for dataset creation, most of them exposes scripting interfaces that allow developers to design custom plug-ins and to automate the generation pipeline. In this context, we propose two geometry-based approaches for synthetic dataset generation tailored to indoor and outdoor MAV scenarios, each validated in real-world robotics competitions.

3 Geometry-Based Synthetic Datasets for MAV Detection

Two primary advantages of geometry-based synthetic datasets are (i) the elimination of manual annotation and (ii) the flexibility in scene generation.

Manual annotation is labor-intensive and prone to inconsistencies arising from varying labeling guidelines. In our approach, object dimensions and positions are extracted via the Blender API to automatically generate bounding boxes and metadata, ensuring both large dataset size and labeling consistency without human intervention. Additionally, the ability to manipulate scene elements, lighting, and camera parameters enhances the representativeness of the generated data. Assets such as 3-D objects and textures were sourced from free repositories like PolyHaven and BlenderKit.

3.1 Landing-Pad Detection Dataset (CBR 2023)

In the 2023 Brazilian Robotics Competition (CBR), the precision-landing challenge of the RoboCup Brazil Flying Robots Trial League required drones to do indoor detection of 1 m × 1 m landing pads, at distances ranging from 1 to 8 m. Figure 2 illustrates the landing pad used for the competition.



Figure 2: Landing pad pattern used in the CBR 2023 competition.

The dataset was generated using Blender's procedural-geometry pipeline, controlled via a Python script. Two techniques were employed:



Figure 3: Images created using technique 1.



Figure 4: Images created using technique 2.

1. **Background-texture variation:** The landing-pad model was rotated and repositioned across varying distances and angles relative to the virtual camera. High-dynamic-range imaging (HDRI) textures were proce-

http://www.imavs.org/

durally rotated and translated to provide contextual variation [5]. Figure 3 presents some samples of the generated images.

2. **Scene-element variation:** The landing pad was placed on various planar terrains with randomized configurations of surrounding objects. Both terrains and objects were procedurally repositioned and rotated. Lighting was modeled with a combination of ambient illumination and directional sources whose color temperatures matched common indoor lights (e.g., fluorescent, halogen, LED). Figure 4 presents samples produced in this process.

Additional frames with close-up views and partial occlusions were rendered to reflect realistic proximity scenarios during landing. All annotations were generated automatically.

3.2 Animal-Detection Dataset (IMAV 2024)

The 2024 International Micro Air Vehicle (IMAV) competition posed an outdoor challenge requiring drones to detect zebra-striped animal markers (Figure 5) from a bird's-eye perspective.



Figure 5: Examples of Top-down Images of Zebra for the Census Task: © Yauheniya Piatrouskaya.

Using the presented technique, Blender scenes of zebra markers embedded in outdoor environments were populated with 3-D objects such as vehicles, vegetation, buildings, and other animals. Lighting conditions simulated the sun's trajectory to introduce temporal variation, and select frames included fog elements to emulate low-visibility conditions. All annotations were generated automatically via scripting.

4 Results

4.1 Hardware and software environment

Training was executed on a workstation with Ubuntu 22.04 LTS (`x86_64`, kernel 6.5), Python 3.10.12, PyTorch 2.7.1 built with CUDA 12.6, and Ultralytics YOLOv8 v8.3.150 [6].

The available accelerator was an NVIDIA GeForce RTX 4060 providing 7 808 MiB of addressable memory.

The YOLOv8-nano backbone was fine-tuned for 200 epochs on $720 \times 720\,px$ images with a mini-batch of 16 using:

```
yolo detect train data=~/data.yaml \
                  model=yolov8n.pt \
                  epochs=200 imgsz=720 \
                  batch=16
```

A total of 1,883 procedurally rendered images were used for training and 401 real photographs (460 annotated landing-pad instances) for validation. The network converged in 17 epochs.

4.2 Performance of the best epoch

The obtained model achieved the following performance indicators:

- Precision: 0.926
- Recall: 0.895
- mAP @0.5: 0.966
- mAP @0.5:0.95: 0.804
- GPU memory: Under 3 GB
- Inference speed: 1.3 ms per frame on the RTX-4060

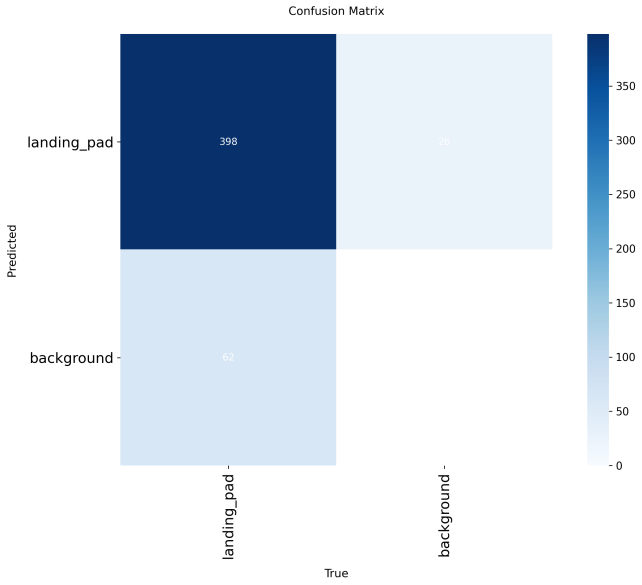


Figure 6: Confusion matrix for the real-image validation set (460 ground-truth instances).

http://www.imavs.org/



Figure 7: Sample predictions on real validation images.

The confusion matrix (Fig. 6) indicates that **87** % of ground-truth objects were correctly detected (TP = 398 of 460), while **6** % of predictions were false positives (FP = 26 of 398 + 26 predictions). The remaining 62 false negatives stem largely from extreme viewing angles and harsh shadows.

Figure 7 illustrates a subset of the real validation set used for quantitative evaluation. The detection results shown were produced by the model trained solely on synthetic images. Despite variability in lighting conditions, viewing angles, shadows, and partial occlusions, the model consistently localized the landing pads with accuracy and robust box placement.

Mean Average Precision (mAP) is adopted as the principal evaluation metric as it is the community standard for ranking detectors in major challenges [7]. Balancing false positives and false negatives, mAP integrates the entire precision–recall curve. Specifically, the COCO protocol computes AP at ten progressively stricter IoU thresholds (0.50 to 0.95 in 0.05 steps) and then averages them, causing the final score to reward both correct class prediction and tight box placement while avoiding the arbitrariness of a single IoU cut-off.

The choice to validate on real images instead of synthetic ones is deliberate. When training and validation come from the same graphics engine, a detector may achieve high scores simply by exploiting rendering regularities; such scores do not guarantee transfer to flight conditions and encourage over-fitting to synthetic artifacts. A real-image validation set—shot under varied lighting, backgrounds and viewing angles—acts as a proxy for deployment and makes the evaluation stricter. Taken together, the high detection metrics (Fig. **??**), low latency and small memory footprint demonstrate that a lightweight network trained exclusively on procedurally generated data generalizes effectively to real imagery and fulfils the accuracy and timing requirements of micro-air-vehicle applications.

The validation used in the CBR 2023 dataset was not applied to the one used in the IMAV 2024 competition. In this case, the effectiveness of the dataset could be determined anecdotally using the results of the actual competition. The Skyrats drone team, using the dataset in question, was able to detect all the zebra markers that the task positioned in the flying area (Fig. 8), with no false positives, even when other animals could be seen in the drone imaging.



Figure 8: 2 of 10 zebras detected at IMAV 2024 (conf. 0.73 & 0.76)

5 Discussion

The dataset creation process revealed key distinctions between synthetic and real-world datasets:

- **Dataset localization** — not specifically related to the task presented in the CBR competition but common in other usages such as rescue missions in outdoors scenarios or the conservation situation presented in IMAV 2024 is the varied possible geographical locations. A real life dataset that tries to offer detection in forests may need the actual location, or something close to it, to be effective in its task. If this is not possible, a synthetic dataset can simulate the location needed provided that the designer knows the region well enough to account for the specifics of it;
- **Automated annotation** — as previously noted, the automated annotation process is a huge benefit of the method developed, skipping the need of an individual to not only label the images correctly, but also to establish and follow the dataset labeling guidelines;
- **Adaptability in new situations** — Another big advantage of the synthetic method provided comes from the adaptability component intrinsic of the method.

http://www.imavs.org/

If needed, as when an aspect of the dataset wasn't representative of real life situations, a new patch in the dataset can be developed relatively easily, without the need to do all the normal steps of data collection and labeling after the fact. This is particularly useful when prototyping a system that may need changes that couldn't be accounted for in the beginning or when mixing real life data with the specific needs of the application being currently developed;

- **Need for qualified individuals** — One of the challenges of the method developed is in how specific the toolset required to create it is. The individuals involved in the project had to have been exposed to the tool (Blender) before, and also needed to have a highly niche knowledge such as using the python API, something not part of the main workflow of the tool;
- **Number of individuals needed** — In a real life situation, the need for multiple pictures might force the dataset to be created by a collective of individuals concurrently so the process is faster. In the synthetic method proposed, even though the individuals need some relevant prior training in the tools involved, the actual number of individuals can be massively decreased, leading to a smaller human footprint on the project.

Table 1: Comparison Between Real-World and Synthetic Dataset Methods

Criterion	**Real-World Dataset**	**Synthetic Dataset**
Localization	Geographically constrained	Region-flexible (designer-dependent)
Annotation time	Manual / semi-automated	Fully automated
Adaptability	Slow, labor-intensive	Rapid and modular
Tool qualification	General labelling tools	3-D tools & scripting required
Team size	Larger teams	Smaller, specialised teams

6 CONCLUSION

This work demonstrates the feasibility and advantages of synthetic dataset generation for MAV applications using procedural geometry and open-source tools. The developed datasets addressed indoor precision-landing and outdoor marker-detection tasks and were validated in robotics competitions. Results show that synthetic datasets are not only viable substitutes but can surpass traditional methods in flexibility, annotation efficiency, and adaptability—despite requiring specific technical expertise.

Next steps to the work presented come with the development of a plug-in inside Blender specialized on helping with synthetic dataset creation. To better validate the results in these new cases, a comparison between the synthetic and a fully real-life based dataset could help elucidate how well both approaches function.

REFERENCES

[1] Yue Hu, Xin-Yu Ye, Yifei Liu, Souvik Kundu, Gourav Datta, Srikar Mutnuri, Namo Asavisanu, Nora Ayanian, Konstantinos Psounis, and Peter Anthony Beerel. Firefly: A synthetic dataset for ember detection in wildfire. *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 3767–3771, 2023.

[2] Nishanth Nandakumar and Jörg Eberhardt. Overview of synthetic data generation for computer vision in industry. In *2023 8th International Conference on Mechanical Engineering and Robotics Research (ICMERR)*, pages 31–35, 2023.

[3] Salehe Erfanian Ebadi, Y. C. Jhang, Alexander Zook, Saurav Dhakad, Adam Crespi, Pete Parisi, Steve Borkman, Jonathan Hogins, and Sujoy Ganguly. Peoplesanspeople: A synthetic data generator for human-centric computer vision. *ArXiv*, abs/2112.09290, 2021.

[4] Matej Arlovic, Davor Damjanovic, Franko Hrzic, and Josip Balen. Synthetic dataset generation methods for computer vision application. In *2024 International Conference on Smart Systems and Technologies (SST)*, pages 69–74, 2024.

[5] Ahmad Rafi, Musstanser Tinauli, and Mohd Izani. High dynamic range images: Evolution, applications and suggested processes. In *2007 11th International Conference Information Visualization (IV '07)*, pages 877–882, 2007.

[6] Rejin Varghese and Sambath M. Yolov8: A novel object detection algorithm with enhanced performance and robustness. In *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*, pages 1–6, 2024.

[7] R. Padilla, S. L. Netto, and E. A. B. da Silva. A survey on performance metrics for object-detection algorithms. In *Proceedings of the 2020 International Workshop on Systems, Signal Processing and their Applications (IWSSIP)*, pages 237–242, 2020.

http://www.imavs.org/

Intelligent vision-based quadrotor path follower

P. Becerra-Martínez[a], D.A. Martínez-Velasco[a], B. Briseño-Tepepa[b] and H. Rodríguez-Cortés[a]*
[a]Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional,
Sección de Mecatrónica, Depto. de Ingeniería Eléctrica,
Ciudad de México, 07360, México
E-mail: {pedro.becerra,dalexis.martinez,hrodriguez}@cinvestav.mx
[b] UPIITA, Instituto Politécnico Nacional
E-mail: bbriseno@ipn.mx

http://www.imavs.org/

Abstract

Autonomous visual navigation in unmanned aerial vehicles (UAVs) can eliminate the need for global positioning systems and predefined maps, favoring more flexible solutions. This work presents an intelligent vision-based path-following controller for a commercial DJI drone. The controller relies on a Convolutional Neural Network (CNN), developed in Python using TensorFlow-Keras, which is trained to predict the desired drone position from a path marked with tape captured by the camera. The training data is obtained manually from supervised flights with automatic labeling. A nonlinear controller forces the drone to achieve the desired position, allowing it to travel the path. The results obtained using the *Shape Context* descriptor method demonstrate that the proposed strategy achieves effective route tracking.

1 Introduction

Computer vision is now a widely used technology; its application has grown significantly since the 2010s, particularly with the development of the Convolutional Neural Network (CNN) AlexNet, which dramatically reduced the error rate in image classification tasks (ILSVRC 2012) [1]. As a result, computer vision is now routinely applied to a wide range of domains, including object detection [2], facial feature recognition [3], medical image analysis [4], crop monitoring [5], and autonomous navigation [6].

In the context of autonomous systems, computer vision provides a rich, passive sensing modality that enables mobile robots to interpret their environment using visual data. This characteristic is especially valuable for Unmanned Aerial Vehicles (UAVs), where autonomous navigation based on computer vision provides a versatile and cost-effective alternative to traditional localization systems, such as the Global Positioning System (GPS), particularly in urban canyons, indoor settings, or where reliance on satellite-based localization is infeasible. The ability to follow visible paths using only onboard cameras minimizes dependency on external infrastructure.

The backbone for solving autonomous navigation tasks using computer vision is the integration of CNNs that enable robust perception of the environment through learned feature hierarchies, are the driving force behind key capabilities in vision-based navigation such as semantic segmentation [7], monocular depth estimation [8], obstacle detection, and visual odometry [9]. These tasks are crucial for real-time autonomous decision-making, enabling UAVs to map their environment, navigate around obstacles, and plan safe trajectories.

CNNs are used in two main navigation paradigms: Mediated perception approaches, where CNN-based perception modules extract high-level features or semantic maps that inform downstream modules for planning and control [10]. Behavioral reflex approaches, where visual inputs are directly mapped to control commands using deep CNNs trained with imitation learning [11] or reinforcement learning [12].

Path planning is one of the key challenges in autonomous navigation. For example, in [13], the authors address the problem of computing minimum-time trajectories for first-person-view (FPV) drones navigating through cluttered environments. Their approach leverages reinforcement learning to generate time-optimal paths while ensuring obstacle avoidance. In the domain of self-driving vehicles, computer vision techniques utilizing onboard cameras are employed for tasks such as image segmentation and object detection. Specifically, geometric shapes of particular colors are identified through visual processing, and this information is used to estimate the appropriate steering angle, enabling the vehicle to follow a predefined path [14].

Nevertheless, several challenges remain in deploying CNNs for autonomous navigation in the wild. These include generalization across diverse environments, handling sensor noise and occlusion, and real-time inference constraints on embedded platforms. Addressing these issues has led to research in domain adaptation [15], uncertainty-aware modeling [16], and hybrid systems that integrate learning-based perception with traditional robotics pipelines [17].

This work focuses on the design of a vision-based path-

*Authors thank the support of Mexican SECIHTI under project Ciencia de Frontera CF-2023-I-551

http://www.imavs.org/

following control system for commercial quadrotors employing a CNN as the perceptual core. The system interprets images captured by a front-facing camera to determine the direction to follow, allowing direct visual tracking of a path defined by a colored tape. The CNN is trained under the mediated perception approach to replace the image processing of a navigation strategy based on the center of mass (CMSS) of the viewed path. The implementation is carried out on a DJI mini 3 pro drone, utilizing a laptop as the processing unit, with a solution developed in Python using TensorFlow-Keras. This configuration aims to maximize portability and leverage resources available in the commercial environment.

2 Real-time video stream processing

2.1 Drone video acquisition

The primary tool for communication with the drone is an Android 14 smartphone with the 5.12.0 version of the DJI Mobile SDK (MSDK) [18] pre-installed on it and connected, via a USB Type-C cable, to the DJI remote controller (RC). The MSDK app was modified by adding two virtual buttons for activating and deactivating Virtual Sticks. The Virtual Sticks allow for commanding the drone's translational velocities and rotational speed around the vertical axis.

Two servers are running on an Acer laptop with an Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, 16.0 GB RAM, 3.01 TB storage with an NVIDIA GPU GeForce GTX 1660 Ti: one for collecting video frames using *MediaMTX* [19], an open-source software that allows for publishing video streams with Real-Time Messaging Protocol (RTMP) over a TCP connection on port 1935. The stream is captured in Standard Definition (SD) landscape orientation (848 x 480 px) from the MSDK app, with a maximum frame rate of 30 FPS. The other one is a TCP-gRPC server through which commands to control the drone are transmitted via Wi-Fi from the laptop to the drone radio controller. This experimental setup was designed in [20]. Consult the official drone documentation [21]

2.2 Image processing with OpenCV library

The video frames are captured in the three-channel RGB color format by a down-facing camera integrated into the drone. Since a particular blue tape defines the desired path, the RGB format is converted to HSV format, which separates color tone (H) from intensity (S) and luminosity (V) [22]. Then, color segmentation is implemented by defining a range between $[100, 150, 50]$ and $[140, 255, 255]$ within the HSV space where the blue color of the tape exists. The frame is then cleaned using morphological operations, such as "close" to fill in the gaps formed by black pixels within the white areas, and "open" to eliminate small isolated pixels, since any noise, gaps, or false spots in the mask directly affect the location of that center. Finally, the pixel values are normalized by dividing the mask by 255, resulting in values between 0 and 1. In practice, this process is equivalent to applying a blue color filter, where ones represent the detected path and zeros correspond to the background. The pseudoalgorithm in 1 illustrates the workflow of this image processing procedure, while Figure 1 presents the effect of the image processing on a image frame.

Algorithm 1 Color Detection using CMSS

A Input: frame, lower_color, upper_color, kernel_size
Output: (x_cmss, y_cmss), mask_clean
B hsv ← ConvertColor(frame, BGR_to_HSV)
C color mask ← InRange(hsv, lower_color, upper_color)
kernel ← Ones(kernel_size, kernel_size)
mask closed ← Morph(mask, MORPH_CLOSE, kernel)
mask clean ← Morph(mask_closed, MORPH_OPEN, kernel)
D mask clean ← mask clean
binary mask ← mask clean / 255
if *Sum(binary mask) == 0* **then**
  return None, mask clean
com ← CenterofMass(binary mask)
if *isnan(com.x) or isnan(com.y)* **then**
  return None, mask clean
else
  x_cmss ← int(com.x) y_cmss ← int(com.y)
  return (x_cmss, y_cmss), mask_clean

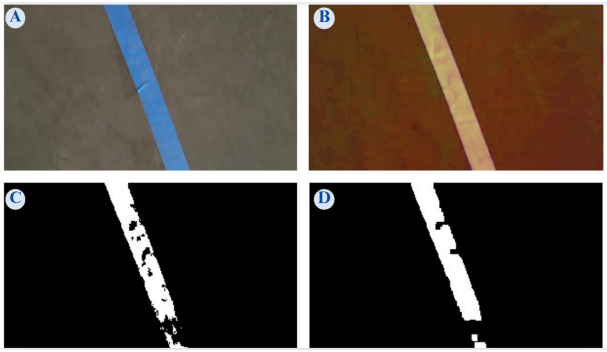


Figure 1: Image processing on a non-augmented sample using Algorithm 1. The input (A), BGR to HSV (B), the color mask (C) and clean mask (D) are shown.

3 Drone model and control

Considering the RC control inputs, the DJI drone flying at a constant altitude can be modeled by the following differential equations [20]. The altitude dynamics are omitted in this model because the drone's onboard control system maintains a nearly constant altitude automatically during flight. This simplification allows the model and controller to focus on horizontal motion, while altitude variations are effectively handled by the drone's internal stabilization.

$$\dot{X} = R_\psi U, \ \dot{R}_\psi = R_\psi S(r) \tag{1}$$

with $X = [\ x \quad y\ ]^\top$ the quadrotor inertial position, $U = [\ u \quad v\ ]^\top$ the quadrotor longitudinal velocity in body

http://www.imavs.org/

coordinates,

$$R_\psi = \begin{bmatrix} c_\psi & -s_\psi \\ s_\psi & c_\psi \end{bmatrix} \in SO(2)$$

the drone attitude with

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

and r the drone angular speed around its vertical axis[1]. Moreover,

$$S(r) = \begin{bmatrix} 0 & -r \\ r & 0 \end{bmatrix} \in \mathfrak{so}(2)$$

with $\mathfrak{so}(2)$ the Lie algebra of $SO(2)$.

At each frame, see Figure 2, it is considered that the inertial reference system is located at the center of the image so that the drone position is $X = 0$. Moreover, the CoM position is defined as the desired drone position X_d and the angle between the $0x^i$ axis and the line that joins the center of the image and the position of the CMSS is equal to ψ.

To ensure compatibility between the image coordinate frame and the drone coordinate frames, all positions are expressed in pixel coordinates. The image coordinate system $0x^v y^v$ is translated to the center by summing to all image measurements $X_c^v = \begin{bmatrix} c_x & c_y \end{bmatrix}^\top$ with $c_x = 424$ and $c_y = 240$, which represent the center of an image of 848×480 pixels.

Using OpenCV the CMSS of the path viewed in the current frame is defined as $X_{CMSS}^v = \begin{bmatrix} cmss_x & cmss_y \end{bmatrix}^\top$, thus

$$X_d^v = X_{CMSS}^v + X_c^v$$

The position error at each frame is defined as

$$\tilde{X} = X_d - X$$

with $X_d = R_{(-\pi/2)} X_d^v$, since at each frame the drone position is assumed to coincide with the image center, it follows that $X = 0$. The angle ψ can be computed as

$$\psi = \arctan\left(\frac{\tilde{y}}{\tilde{x}}\right) \tag{2}$$

The control objective is stated as follows.

Control objective. Assume that X_{CoM} is available. Design U and r such that between image frames $\tilde{X}$ and ψ approach zero. The control objective is satisfied with the following controller. For the translational position

$$U = R_\psi^\top \tilde{X} \tag{3}$$

For the angular position the controller is designed following [23]. The desired attitude is $\psi_d = 0$ so that $R_{\psi_d} = I$. The attitude error is $\tilde{R}_\psi = R_\psi$. Thus,

$$r = -k_r P_a(R_\psi)^\vee \tag{4}$$

with

$$P_a(\tilde{R}_\psi) = \frac{1}{2}\left(\tilde{R}_\psi - \tilde{R}_\psi^\top\right) \in \mathfrak{so}(2)$$

and $(\cdot)^\vee : \mathfrak{so}(2) \to \mathbb{R}$.

[1] In the following, $s_\psi = \sin(\psi)$ and $c_\psi = \cos(\psi)$.

4 Navigation strategies

Once each frame from the video stream is processed, it is used to define a control strategy that guides the drone along the path. The intelligent control strategy was developed as follows. First, the processed image is employed to determine a point over the path that the drone must follow. This point is computed using OpenCV commands such as the CMSS of the detected path with respect to the image center. A yaw angle is also determined as the angle between the line that joins the CMSS location and the center of the image and the vertical axis of a reference frame attached to the image frame. The point location and angle are fed into the controller, which drives the drone to travel along the path. Then, using the data from this navigation strategy, a CNN is trained to determine the point location and angle that are fed to the controller.

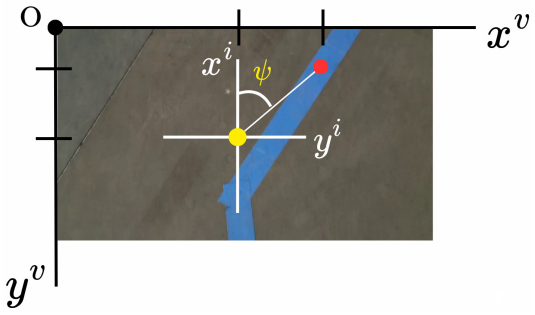


Figure 2: Inertial frame $0x^i y^i$ origin (yellow point). Path CMSS X_d (red point). Yaw angle ψ.

4.1 Navigation without the CNN

The pseudoalgorithm 1 ends with all the information to implement the controller (3), (4). In the results section is shown that the drone is able to track the path.

4.2 Navigation with the CNN

Now, data from four flights of navigation without the CNN were saved; this data consists of an image frame labeled with the corresponding CMSS location and saved to a CSV file organized in three columns. This data was used to train the CNN and compute the path based on CMSS location predictions.

Data collection

From the navigation experiments without CNN, one frame out of every six frames was saved, resulting in a total of 2065 samples organized into four folders: two corresponding to clockwise (CW) route tracking and two to counterclockwise (CCW). This division was necessary because, in practice, the route is not always followed in exactly the same way. By structuring the dataset in this way, we ensured that both directionality and variability in the path execution were adequately represented for training and evaluation.

To strengthen CNN training for color and shape detection in scenarios where luminosity may vary, data augmentation was employed. For each image sample, five

versions of it were generated, two to vary the brightness and one for contrast, using the following relationship [24]:

$$g(i,j) = \alpha \cdot \mathbf{f(i,j)} + \beta$$

where $f(i,j)$ and $g(i,j)$ denote the original and new pixel values at position (i,j), respectively. The parameter $\alpha \in (0,1)$ adjusts contrast, while β controls brightness: $\beta < 0$ darkens the image, and $\beta > 0$ increases brightness.

In the fourth and fifth images the gamma value was modified according to the formula [24]:

$$O = \left(\frac{\mathbf{I}}{255}\right)^{\gamma} \cdot 255 \tag{5}$$

where O are the mapped output values from the image, I are the input values; for $0 < \gamma < 1$ light areas appear darker and $\gamma > 1$ darker areas appear lighter. The CMSS position label was only copied to the extra five variations of each frame.

After data augmentation the dataset has 38,778 image frames and labels. Figure 3 shows the augmented data and the original frame.

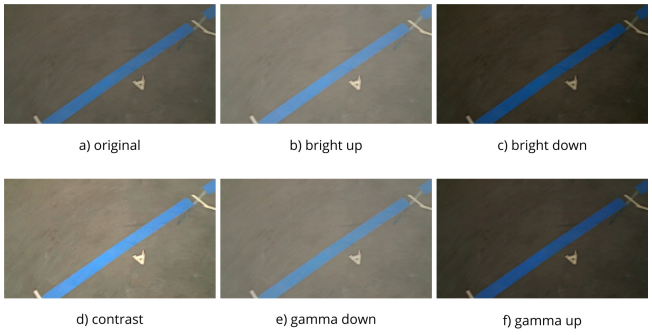


Figure 3: Sample of the dataset after the data augmentation.

4.3 CNN

The CNN was trained with TensorFlow. The input for model is a dataset binarized and resized to 300x169 px, with a single channel (grayscale) normalized in the range $[0,1]$. The structure of the network is composed of three convolutional blocks, see Table 1, with ReLU activation and max-pooling, followed by dense layers that perform the regression of two continuous values, the coordinates of the CMSS X_{CMSS}.

From the augmented dataset, only 20,000 samples were selected for training, due to the size of the processing unit memory in the GPU which is 6 GB. The maximum number of samples was determined by trial and error. The 80% was used for training and the rest for validation. The model was trained for a maximum of 100 epochs with a batch of 16 and the *Adam* optimizer with the default learning rate. Regarding the training errors, the Mean Square Error (MSE) is used as a loss function while the Mean Absolute Error (MAE) is used as an evaluation metric that represents the average deviation between model predictions and actual values. To

	Layer	Details
0	Conv2D (Input)	32 filters, 3×3 kernel, ReLU activation
1	MaxPooling2D	2×2 pool size
2	Conv2D	64 filters, 3×3 kernel, ReLU activation
3	MaxPooling2D	2×2 pool size
4	Conv2D	128 filters, 3×3 kernel, ReLU activation
5	MaxPooling2D	2×2 pool size
6	Flatten	Converts 3D tensor to 1D
7	Dense	128 units, ReLU activation
8	Dense (Output)	2 units (linear activation for regression)

Table 1: CNN Architecture.

avoid overfitting, *Early Stopping* strategies were implemented, and the model was saved with the lowest loss in validation using *Model Checkpoint*. The combination of software and library versions that allowed running the Nvidia GPU for training correctly is presented in Table 2.

Software/Library	Details
Operating system	Windows 11
Python	3.11.4
TensorFlow	2.18.0
Sklearn	1.3.0
NumPy	1.24.3
CUDA Toolkit	12.7
cuDNN	8.6
GPU	NVIDIA GTX 1660 Ti
NVIDIA driver	566.36
VSCode	1.102.1
Anaconda	2.6.3

Table 2: Computational resources for CNN training.

5 Results

5.1 Trained CNN

The training process took 42 epochs to complete, during which the MAE and the loss function were monitored for training and validation sets.

Figure 4 shows the evolution of the MAE. A continuous decrease is observed in the training set, from 0.0330 to 0.0092, indicating that the model undergoes progressive learning from the dataset. The validation set decreases towards an oscillation phase between 0.0169 and 0.0207. This variability suggests that the model reached a saturation point in its capacity to generalize. The model achieved a final MAE of 0.0154 over normalized coordinates, equivalent of $\approx$ 13 px in X and $\approx$ 7 px in Y, corresponding to an average Euclidean distance of $\approx$ 15 px.

Figure 5 corresponds to the loss function (MSE) during the training process. From epoch 10 onwards, training loss decreases until it stabilizes around 0.0006. In the validation set, it stabilizes without showing relevant changes, which suggests that the model does not incur significant overfitting and maintains its generalization capacity throughout training. The final loss (MSE) was 0.0012, whose root mean (RMSE) corresponds to an er-

http://www.imavs.org/

ror of ≈ 29 px in X and ≈ 17 px in Y, with an average Euclidean distance of ≈ 34 px on the validation set. The model learned around epoch 15–20, suggesting that continuing training beyond this point offers little improvement. Model performance was confirmed by real-world testing.

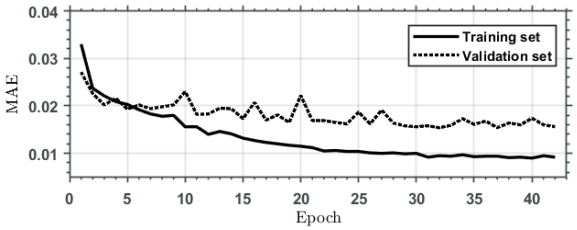


Figure 4: MAE vs Epoch graph.

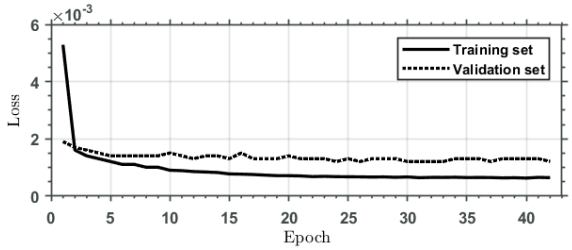


Figure 5: Loss (MSE) vs Epoch graph.

5.2 *Experimental tests*

Two runs were performed using the CMSS calculation and two more using the predicted values from the model, CW and CCW. Figure 6 and Figure 7 were obtained from these runs, where the trajectories were plotted using data from an *Optitrack* system.

To verify the following of the real path, the *Shape Context* algorithm was implemented, which is scale invariant and reflects the structural difference between shapes [25], estimating correspondences between points in the geometry paths traced by both, predictions from the CNN and CMSS. The result indicates that with the CNN prediction method with a shape context distance of 64.6561 (CCW) and 123.3177 (CW) it has better similarity with the original trajectory, while the shape context distance of the CMSS was 71.5434 (CCW) and 133.6464 (CW) making it further away in geometric structure.

6 Conclusions

This work presented the design of a control algorithm calculated from CNN predictions of coordinates so that a commercial DJI drone can navigate autonomously on a reference, using freely available software tools that can be used for real-time image processing with computer vision. The system was validated through experiments under real-world conditions. In the initial stage, tracking issues were identified due to light reflections in areas on the floor where the blue tape was placed. This led to increasing the training dataset by varying the illumination range of the images already taken. The capabilities of the proposed system were tested and met the objective, as can be seen in the comparative graphs in Figure 6 and Figure 7 of the experimental routes and the difference calculated with the Shape context algorithm. The result is that the predictions of a trained model combined with computer vision can be an alternative for solving real-time autonomous navigation tasks. To complete the experiment, a full analysis of the CNN used is currently underway. Consideration is being given to strengthening the system by using deeper networks such as CNN+LSTM, thus capturing the temporal evolution of the tracking to provide early predictions. This would open up the possibility of integrating more advanced predictive controllers.

Suplementary material at:`https://youtube.com/playlist?list=PLIFqXxbt6YQT_ZDaQDj3KNkFF3OEGUaGR&si=-vMcyr-04_M1mLYf`

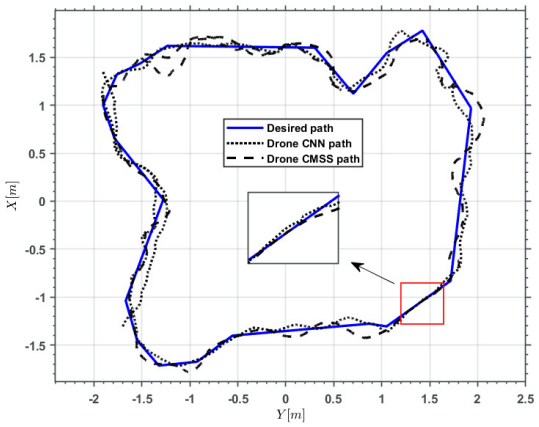


Figure 6: Path traveled CW.

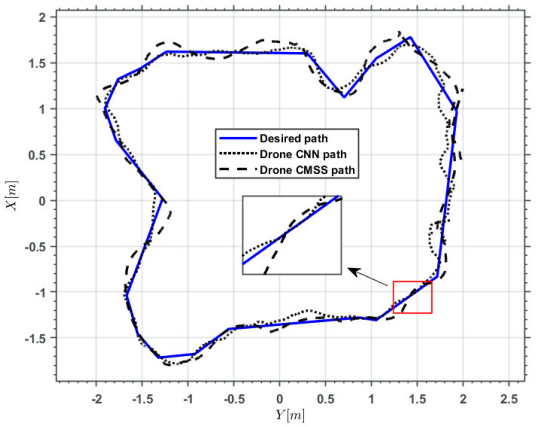


Figure 7: Path traveled CCW.

http://www.imavs.org/

http://www.imavs.org/

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[2] Y. Xiao, A. Jiang, J. Ye, and M.-W. Wang, "Making of night vision: Object detection under low-illumination," *IEEE Access*, vol. 8, pp. 123075–123086, 2020.

[3] K.-C. Liu, C.-C. Hsu, W.-Y. Wang, and H.-H. Chiang, "Real-time facial expression recognition based on cnn," in *2019 international conference on system science and engineering (ICSSE)*, pp. 120–123, IEEE, 2019.

[4] J. Olveres, G. González, F. Torres, J. C. Moreno-Tagle, E. Carbajal-Degante, A. Valencia-Rodríguez, N. Méndez-Sánchez, and B. Escalante-Ramírez, "What is new in computer vision and artificial intelligence in medical image analysis applications," *Quantitative imaging in medicine and surgery*, vol. 11, no. 8, p. 3830, 2021.

[5] D. Story and M. Kacira, "Design and implementation of a computer vision-guided greenhouse crop diagnostics system," *Machine vision and applications*, vol. 26, no. 4, pp. 495–506, 2015.

[6] D. Lee, G. Kim, D. Kim, H. Myung, and H.-T. Choi, "Vision-based object detection and tracking for autonomous navigation of underwater robots," *Ocean Engineering*, vol. 48, pp. 59–68, 2012.

[7] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440, 2015.

[8] C. Godard, O. M. Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6602–6611, 2017.

[9] S. Wang, R. Clark, H. Wen, and N. Trigoni, "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, p. 2043–2050, IEEE, May 2017.

[10] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2722–2730, 2015.

[11] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," 2016.

[12] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8248–8254, 2019.

[13] R. Penicka, Y. Song, E. Kaufmann, and D. Scaramuzza, "Learning minimum-time flight in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7209–7216, 2022.

[14] S. Shafique, S. Abid, F. Riaz, and Z. Ejaz, "Computer vision based autonomous navigation in controlled environment," in *2021 International Conference on Robotics and Automation in Industry (ICRAI)*, pp. 1–6, IEEE, 2021.

[15] Y. Zhang, P. David, H. Foroosh, and B. Gong, "A curriculum domain adaptation approach to the semantic segmentation of urban scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 8, pp. 1823–1841, 2020.

[16] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?," *Advances in neural information processing systems*, vol. 30, 2017.

[17] F. Atas, G. Cielniak, and L. Grimstad, "Elevation state-space: Surfel-based navigation in uneven environments for mobile robots," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5715–5721, 2022.

[18] DJI Developers, "Dji mobile sdk documentation - introduction." `https://developer.dji.com/mobile-sdk/documentation/introduction/index.html`, 2025.

[19] GitHub Repository:bluenviron/mediamtx, "mediamtx: Ready-to-use rtsp server and media proxy." `https://github.com/bluenviron/mediamtx`, 2025.

[20] M. Martínez-Ramírez, M. Trujillo-Flores, X. Shao, J. G. Romero, and H. Rodríguez-Cortés, "A collision avoidance strategy for commercial quadrotors," in *2025 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 988–993, 2025.

[21] DJI, "Mini 3 pro - product support." `https://www.dji.com/mx/support/product/mini-3-pro`, 2025.

[22] OpenCV Documentation, "Changing colorspaces — opencv documentation." `https://docs.opencv.org/4.x/df/d9d/tutorial_py_colorspaces.html`, 2025.

[23] H. Rodríguez-Cortés and M. Velasco-Villa, "A new geometric trajectory tracking controller for the unicycle mobile robot," *Systems & Control Letters*, vol. 168, p. 105360, 2022.

[24] OpenCV Documentation, "Basic linear transforms — opencv documentation." `https://docs.opencv.org/4.x/d3/dc1/tutorial_basic_linear_transform.html`, 2025.

[25] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 4, pp. 509–522, 2002.

Autonomous UAV Landing with Visual Servoing and Semi-Markov Decision Processes

Fabián González-Tejeda, Brenda Ivette García-Maya*, and Hernán Abaunza
Tecnologico de Monterrey, School of Engineering and Sciences

http://www.imavs.org/

Abstract

This paper presents an innovative approach to autonomous UAV landing on moving platforms by integrating a Semi-Markov Decision Process (SMDP) framework with a visual servoing control strategy. The proposed method models the UAV's position relative to the landing target using a semi-Markov state space, accounting for varying visibility conditions and the precise positioning required for a successful landing. By incorporating a reward function, the SMDP framework enables the UAV to evaluate trade-offs between different actions and dynamically optimize its trajectory. This approach enhances the UAV's ability to adapt to platform movement, handle uncertainty, and improve landing performance in unpredictable environments. The contribution of this work lies in the novel application of SMDPs combined with visual servoing, offering a robust, adaptable solution to the challenges of autonomous landing in real-world, dynamic conditions.

1 Introduction

Unmanned Aerial Vehicles (UAVs) have rapidly advanced in response to the growing demand for real-time applications in dynamic environments. Consequently, autonomous landing systems have become essential for various tasks [1], including package delivery [2], search and rescue [3], and aerial inspections requiring timely decisions [4].

These systems depend on advanced algorithms and sensors to perform precise, reliable landings. Among state-of-the-art methods, visual servoing [5, 6] stands out as a popular approach [7, 8], providing accurate control by processing visual feedback to guide the UAV during descent. However, visual servoing faces significant challenges related to environmental variability, occlusions, sensor inaccuracies, and computational limitations, demanding more sophisticated strategies for robust operation [9, 10].

*Corresponding author
F. González-Tejeda: A01634223@tec.mx, ORCID: 0009-0002-1597-6244,
B.I. García-Maya: brenda.gmaya@tec.mx, ORCID: 0000-0002-9945-3784.
H. Abaunza: habaunza@tec.mx ORCID: 0000-0001-5325-730X.
Tecnologico de Monterrey, School of Engineering and Sciences
Guadalajara Campus, Av. General Ramón Corona 2514, 45201 Zapopan, Jalisco, México.

To address these issues, Kalman filters have been widely adopted to model and manage uncertainty during landing maneuvers [11]. While effective for filtering noise and estimating states, they may struggle under highly unpredictable conditions—particularly when vision-related problems arise, such as lighting changes, occlusions, target motion, and UAV inclination variations, especially without gimbal stabilization.

To improve robustness under uncertainty, alternative probabilistic frameworks like Markov chains have been explored. By modeling the landing process as discrete states with transition probabilities, Markov chains enable detailed analysis of possible outcomes based on UAV actions. Extending this concept, Markov Decision Processes (MDPs) [12] introduce decision-making into the framework, allowing UAVs to handle stochastic environments by associating actions with rewards and transition probabilities [13].

The use of Markov models represents a significant step toward overcoming UAV landing challenges, improving reliability and adaptability in uncertain scenarios [14]. However, standard Markov processes assume sojourn times follow geometric or exponential distributions, limiting their flexibility. Semi-Markov systems address this limitation by allowing any probability distribution for sojourn times [15].

This work aims to improve autonomous UAV landing on moving platforms by integrating a Semi-Markov Decision Process (SMDP) framework with visual servoing. The SMDP models the UAV's state relative to the target, accounting for varying visibility and precise positioning needed for target locking and landing. A reward function guides the UAV to evaluate trade-offs and select actions that maximize landing success probability. This approach enables adaptation to platform motion, uncertainty management, and trajectory optimization. The main contribution is this novel combination of SMDP and visual servoing, offering a robust, adaptable solution for real-world UAV landing scenarios.

The remainder of this paper is organized as follows: Section 2 describes the proposed methodology; Section 3 details the visual servoing control strategy; Section 4 presents the SMDP framework for autonomous decision-making; Section 5 discusses the experimental validation; and Section 6 provides conclusions and future perspectives.

2 Methodology

The proposed solution integrates **Visual Servoing** and **Semi-Markov Decision Processes (SMDPs)** to train a reliable UAV landing system, as illustrated in Figure 1. **Visual**

http://www.imavs.org/

Servoing is responsible for state estimation, providing real-time information about the relative position of the landing platform. **SMDPs**, on the other hand, enable decision-making by modeling the system's behavior over time.

The system processes two main inputs:

1. **Real-time platform position (Input 1a):** The UAV's onboard camera captures the relative position of the landing platform. This information is converted into discrete states representing the platform's location in relation to the UAV.

2. **Experimental flight data (Input 1b):** A dataset is collected from previous flights, recording the observed states, actions taken, and corresponding outcomes. This historical data is used to estimate the probabilities of different system behaviors over time, forming a transition probability matrix.

Using this probability matrix, the system evaluates the expected rewards for different actions in each state. It computes the optimal action the UAV should take to maximize its chances of a successful landing. Once computations are completed (Steps 3, 4, 5, and 6 in Figure 1), the drone controller (acting as the pilot) selects the most effective decision for the given scenario.

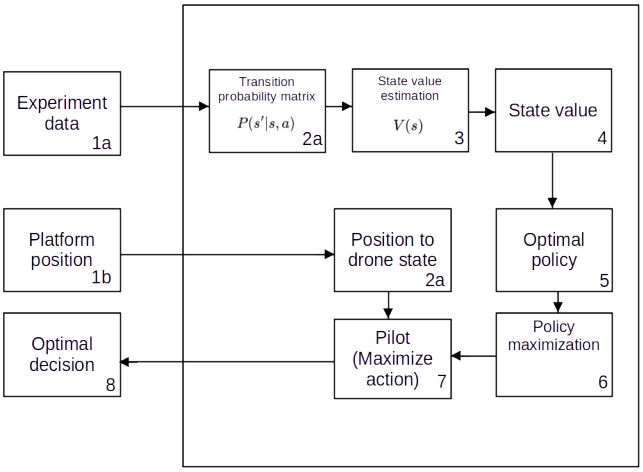


Figure 1: UAV landing decision process algorithm diagram

This structured approach allows the UAV to dynamically adapt its landing strategy based on real-time observations and learned experience, improving accuracy and reliability.

The relative pose of the moving platform is estimated by detecting visual markers (e.g., QR codes) from the camera feed. The visual data is interpreted and converted into a series of discrete states (detailed in Section 4), which are used to ensure real-time awareness of the system's status.

Then, a SMDP, models the problem as a sequence of states, actions, and rewards. The state space captures the UAV's relative position to the target and the status of the landing platform, with terminal states representing either a successful landing or a failed attempt. The action space defines the UAV's possible maneuvers, such as adjusting circular velocity, moving in specific directions, or centering the target based on visual feedback.

State transition probabilities account for system uncertainties, representing the likelihood of moving from one state to another after performing a specific action. The reward function is designed to promote behaviors that guide the UAV toward the target while penalizing inefficient actions.

The objective is to find an optimal policy $\pi^*(s)$, which is a function that maps each state s to an optimal action a, maximizing the expected cumulative reward:

$$\pi^*(s) = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right],$$

where γ is the discount factor that determines the importance of future rewards. The reward function R is carefully crafted to minimize the landing error and ensure that the UAV makes decisions that lead to a safe and accurate landing.

The optimal policy is computed using linear programming techniques. These algorithms enable the UAV to learn the optimal policy through trial and error, updating its actions based on the observed rewards. The integration of Visual Servoing and SMDPs allows the system to adapt dynamically to the changing environment, making real-time adjustments based on the current state.

The system operates in a feedback loop, where Visual Servoing continuously estimates the platform's position and the SMDP evaluates the state and selects an optimal action. This loop ensures that the UAV can adjust its trajectory in response to platform motion and other uncertainties, improving the likelihood of a successful landing.

3 UAV MODELING AND VISUAL SERVOING STRATEGY

The UAV's dynamics are governed by:

$$m\ddot{\mathbf{p}} = \mathbf{R}(\mathbf{q}) \cdot \mathbf{f_b} - m\mathbf{g}, \tag{1}$$

$$\mathbf{J}\dot{\boldsymbol{\omega}} = \boldsymbol{\tau} - \boldsymbol{\omega} \times (\mathbf{J}\boldsymbol{\omega}), \tag{2}$$

where $m \in \mathbb{R}$ is the UAV mass, $\ddot{\mathbf{p}} \in \mathbb{R}^3$ its acceleration (inertial frame), $\mathbf{R}(\mathbf{q}) \in \mathbb{R}^{3\times3}$ the rotation matrix from quaternion $\mathbf{q} \in \mathbb{H}$, $\mathbf{J} \in \mathbb{R}^{3\times3}$ the inertia matrix (body frame), and $\boldsymbol{\omega} = [\omega_x \ \ \omega_y \ \ \omega_z]^T$ the angular velocity. Gravity is given by $\mathbf{g} = [0\,,\, 0\,,\, -9.81]^T$ m/s^2.

The internal control system operates as follows:

- **Inputs:** Desired translational velocity $\mathbf{v_d} = [v_{d_x} \ \ v_{d_y} \ \ v_{d_z}]^T$ and yaw angular velocity $\omega_{z_d} \in \mathbb{R}$.
- **Outputs:** Thrust force $f_b \in \mathbb{R}$ and control torque $\boldsymbol{\tau} \in \mathbb{R}^3$ (body frame).

http://www.imavs.org/

Assumption 1 *The internal controller (provided by the manufacturer) ensures that* $\mathbf{v} \rightarrow \mathbf{v_d}$*, and* $\boldsymbol{\omega} \rightarrow [0\,,\,0\,,\,\omega_{z_d}]^T$*, where* $\mathbf{v} = [v_x \;\; v_y \;\; v_z]^T = \dot{\mathbf{p}}$ *is the translational velocity (inertial frame) and* $\boldsymbol{\omega}$ *the angular velocity (body frame).*

3.1 Visual Servoing for Autonomous Landing

The use of Image-Based Visual Servoing (IBVS) in UAV control relies on image feedback to achieve precise positioning [16]. Unlike position-based methods, IBVS operates directly in the image plane, reducing dependency on accurate 3D scene reconstruction. This approach is particularly effective for tasks such as landing, object tracking, and navigation in dynamic environments. The operational scheme of **Image-Based Visual Servoing (IBVS)** for multirotor UAVs is illustrated in Figure 2, highlighting the feedback loop between visual data and control commands. The diagram represents the relationship between the **image plane** and the **drone position plane**. The UAV moves along the x, y, and z axes, representing the spatial coordinates in which it can maneuver freely.

The **image plane** is a two-dimensional (2D) representation of the captured scene, where positions are defined in pixel coordinates (u, v). The observed image is composed of pixel coordinate sets, with the image center defined as (u_0, v_0). To align the image center with a target, we calculate the distance between the center and any point in the image plane, denoted as $p = [u, v]$.

Once the desired trajectory is determined in the image plane, mapping it to the drone's movement requires a transformation into the **position plane**, where coordinates are expressed as $p = [x, y, z]$. This transition between the image plane and the position plane is achieved using the **Jacobian matrix**, which establishes the mathematical relationship between pixel displacement and UAV motion. More details on this transformation are provided in the following sections.

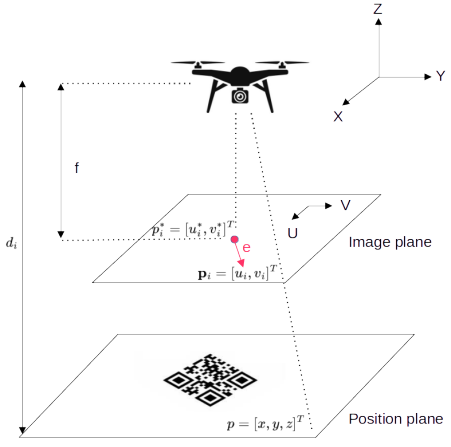


Figure 2: IBVS control scheme for a multirotor UAV, illustrating image plane error correction for precise target alignment.

3.1.1 Error Definition

The error in IBVS is defined as the distance difference between an actual point and our desired position:

$$\mathbf{e} = \mathbf{p}_i - \mathbf{p}_i^* \tag{3}$$

where: $\mathbf{p}_i = [u_i, v_i]^T$ denotes the observed image feature position, and $\mathbf{p}_i^* = [u_i^*, v_i^*]^T$ represents the desired image feature position.

The image position is represented in the image frame, described by the camera position. The transformation with the jacobian matrix help transform the desired position in the camera frame to the a desired velocity for the UAV.

3.1.2 Pixel Velocity Control Law

To understand the pixel velocity of an image it is important to compute the relationship between the pixel velocity and the camera velocity, to achieve this we multiple the camera velocity by the image Jacobian matrix which is given by:

$$\dot{\mathbf{p}}_i = J\boldsymbol{\nu} \tag{4}$$

where:

- $\dot{\mathbf{p}}_i = [\dot{u}, \dot{v}]^T$: pixel velocity.
- $\boldsymbol{\nu} = [\mathbf{v}^T, \boldsymbol{\omega}^T]^T = [v_x, v_y, v_z, \omega_x, \omega_y, \omega_z]^T$: UAV translational and rotational velocities.
- J: Jacobian matrix relating UAV velocities to image feature velocities:

$$J = \begin{bmatrix} -\frac{f_x}{d_i} & 0 & \frac{u_i}{d_i} & \frac{u_i v_i}{f_x} & -(f_x + \frac{u_i^2}{f_x}) & v_i \\ 0 & -\frac{f_y}{d_i} & \frac{v_i}{d_i} & \frac{v_i^2}{f_y} & -\frac{u_i v_i}{f_y} & -u_i \end{bmatrix} \tag{5}$$

where f_x, f_y are the focal lengths, d_i is the depth of the feature, and u_i, v_i are the coordinates of the feature in the image plane.

3.1.3 Control Law

Considering Assumption 1, a control law that computes the desired camera velocity can be expressed as:

$$\boldsymbol{\nu}_d := -\lambda J^{\dagger}\mathbf{e} \tag{6}$$

where λ denotes a positive gain, while $J^{\dagger}$ is the pseudo-inverse of the Jacobian matrix. Introducing (6) into (4) yields

$$\dot{\mathbf{p}}_i = -J\lambda J^{\dagger}\mathbf{e} = -\lambda\mathbf{e}, \tag{7}$$

considering $\dot{\mathbf{p}}_i \approx \dot{\mathbf{e}} = -\lambda\mathbf{e}$ results in the convergence of $\mathbf{e} \rightarrow 0$.

Considering Assumption 1, then $\omega_x \approx \omega_y \approx 0$, therefore, the desired velocities for the UAV are computed as

$$\begin{bmatrix} v_{d_x} \\ v_{d_y} \\ v_{d_z} \\ \omega_{d_z} \end{bmatrix} = -\lambda \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} J^{\dagger}\mathbf{e}. \tag{8}$$

http://www.imavs.org/

4 SEMI-MARKOV DECISION PROCESS FOR AUTONOMOUS LANDING

Before introducing semi-Markov Decision Processes (SMDP), we will present some necessary mathematical concepts for their understanding, such as the concept of Markov chains.

4.1 Markov chains

A Markov chain is a mathematical model that describes a sequence of random variables where the probability of transitioning to the next state depends only on the current state and not on the previous states (the Markov property). Formally, let $\mathbb{N} := \{0, 1, 2, ..\}$ be the set of natural numbers. The sequence $\{\mathcal{X}_t\}_{t\in\mathbb{N}}$ which takes values in state space S is a Markov chain if it satisfies the following equation:

$$\begin{aligned} &P(\mathcal{X}_{t+1} = j \mid \mathcal{X}_t = i, \mathcal{X}_{t-1} = \cdot, \ldots, \mathcal{X}_0 = \cdot) \\ &= P(\mathcal{X}_{t+1} = j \mid \mathcal{X}_t = i), \qquad \forall\ i, j \in S. \end{aligned}$$

Markov chains are essential for modeling dynamic systems where the next state depends only on the current state. However, as discussed in the introduction, the primary limitation when utilizing Markov chains for system modeling lies in the sojourn time in a state. The sojourn time is a random variable that must be modeled by a geometric distribution (in the discrete case) or by an exponential distribution (in continuous time). To generalize this hypothesis in this article we propose a SMDP to achieve the landing aircraft decision. Before introducing the decisions equations we shall introduce semi-Markov processes.

4.2 semi-Markov processes

Let us consider a complete probability space $(\Omega, \mathcal{F}, \mathbb{P})$ on which all processes and random variables are defined. Let $\{\mathcal{X}_t\}_{t\in\mathbb{N}}$ be a stochastic process which takes values in state space S. Let us also define by $\{\tau_n\}_{n\in\mathbb{N}}$ the consecutive time instants when $\{\mathcal{X}_t\}_{t\in\mathbb{N}}$ shifts its state. By definition $\tau_0 := 0$ and

$$\tau_{n+1} := \inf\{t > \tau_n : \mathcal{X}_t \neq \mathcal{X}_{\tau_n}\},\ \ n \geq 0.$$

By convention $\inf \varnothing = \infty$. Observe that sequence $\{\tau_n\}_{n\in\mathbb{N}}$ represents the renewal or jump points of process $\{\mathcal{X}_t\}_{t\in\mathbb{N}}$. We shall denote by $\{B_t\}_{t\in\mathbb{N}}$ the backward process of process $\{\mathcal{X}_t\}_{t\in\mathbb{N}}$. Process $\{B_t\}_{t\in\mathbb{N}}$ is mathematically defined as follows

$$B_t := t - \max\{\tau_n : \tau_n \leq t\} \ \text{ and } B_t := t, \ \text{ if } \tau_1 > t.$$

Speaking generally, the backward process models the time spent in the current state since the last state change.
We define $N(t) := \max\{n \in \mathbb{N} : \tau_n \leq t\}$ as the counting process of the number of jumps in the interval time $[1, t]$ of process $\{\mathcal{X}_t\}_{t\in\mathbb{N}}$. Let us denote by $\{\mathcal{Y}_n\}_{n\in\mathbb{N}}$ a process which records $\{\mathcal{X}_t\}_{t\in\mathbb{N}}$ at jump points $\{\tau_n\}_{n\in\mathbb{N}}$, i.e., $\mathcal{Y}_n = \mathcal{X}_{\tau_n}$ or equivalent $\mathcal{X}_t = \mathcal{Y}_{N(t)}$. We represent by $\{T_n\}_{n\in\mathbb{N}}$ the process that models the time interval between two consecutive jumps. By convention $T_0 := \tau_0$ and $T_{n+1} := \tau_{n+1} - \tau_n$, $n \in \mathbb{N}$. If the following condition is satisfied

$$\begin{aligned} \mathbb{P}(\mathcal{Y}_{n+1} = j,\ T_{n+1} = k \mid \mathcal{Y}_0 = \cdot, \ldots, \mathcal{Y}_n = i;& \\ \tau_0 = \cdot, \ldots, \tau_n = \cdot)& \\ = \mathbb{P}(\mathcal{Y}_{n+1} = j,\ T_{n+1} = k \mid \mathcal{Y}_n = i)& \quad (9) \end{aligned}$$

then process $\{\mathcal{X}_t\}_{t\in\mathbb{N}}$ is a semi-Markov chain (SMC) and process $\{\mathcal{Y}_n\}_{n\in\mathbb{N}}$ is know as the embedded Markov chain of process $\{\mathcal{X}_t\}_{t\in\mathbb{N}}$. From now on to make the notation less cumbersome, instead of writting $\{\mathcal{X}_t\}_{t\in\mathbb{N}}$ we will write $\{\mathcal{X}_t\}$ and apply this simplification to all processes.
If the right-hand side of Equation (9) does not depend on n then $\{\mathcal{X}_t\}$ and $\{\mathcal{Y}_t\}$ are time homogeneous. In this article we shall consider a homogeneous discrete-time semi-Markov process $\{\mathcal{X}_t\}$. A discrete-time semi-Markov process is also referred to as a semi-Markov chain (SMC). A SMC is fully specified by its state space, in this case it will be denoted by S and, its semi-Markov kernel $\mathbf{K}(t) = \{K_{ij}(t);\ \ i, j \in S, t \in \mathbb{N}\}$ where

$$K_{ij}(t) := \mathbb{P}(\mathcal{Y}_{n+1} = j, T_{n+1} = k \mid \mathcal{Y}_n = i), \quad k \geq 0,\ t \in \mathbb{N}. \tag{10}$$

The cumulative semi-Markov kernel of SMC $\{\mathcal{X}_t\}$ is defined by

$$\begin{aligned} Q_{ij}(t) \ &:= \ \mathbb{P}(\mathcal{Y}_{n+1} = j, T_{n+1} \leq t \mid \mathcal{Y}_n = i) \\ &= \ \sum_{k=0}^{t} K_{ij}(k),\ \ i, j \in S, t \in \mathbb{N}. \end{aligned}$$

The cumulative distribution function of the sojourn time in state $i \in S$ is defined by

$$H_i(t) := \sum_{k=0}^{t} \sum_{j\in S} K_{ij}(k)$$

The conditional cumulative distribution of the sojourn time is

$$\begin{aligned} F_{ij}(t) := \mathbb{P}(T_{n+1} \leq t \mid J_n = i, J_{n+1} = j)& \\ = \begin{cases} \frac{Q_{ij}(t)}{p_{ij}}, & \text{if } p_{ij} \neq 0, \\ 1, & \text{if } p_{ij} = 0. \end{cases}& \quad (11) \end{aligned}$$

The primary distinction between Markov and semi-Markov discrete-time processes lies in the distribution function $F_{ij}(t)$. For a Markov chain, this function follows a geometric distribution with a success parameter of $1 - P(i, i)$, where P represents the Markov transition probability matrix. Conversely, in a semi-Markov chain, the distribution function $F_{ij}(t)$ may be represented by any discrete-time distribution.
Throughout this article, note that we employ the index $t \in \mathbb{N}$ to denote calendar time, while the index $n \in \mathbb{N}$ is used for the renewal points of SMC $\{\mathcal{X}_t\}$.

http://www.imavs.org/

In this subsection, we provided a summary of the fundamental background on a SMC. In the following subsection, we will present the Markov Decision Process (MDP). After that we shall generalize the MDP into SMDP.

4.3 Markov Decision Process (MDP)

A MDP is a mathematical framework used for modeling decision-making situations where outcomes are partly random and partly under the control of a decision maker. It consists of the following components:

- States S: A set of possible situations or configurations the system can be in.
- Actions A: A set of possible actions the decision maker can take from a state.
- Transition probabilities between states: The probability of transitioning from one state to another.
- Rewards R: A numerical value received after taking an action in a state, representing the immediate benefit or cost of that action.
- Discount factor γ: A constant number ($0 \leq \gamma \leq 1$). If $\gamma \approx 1$ the system prioritiza future rewards by contrast if $\gamma \approx 0$ the system prioritize immediate rewards.
- Policy π: A strategy or a plan that defines the action to be taken in each state.

The goal of a MDP is to find an optimal policy that maximizes the expected value of long-term rewards. To achieve this goal, Bellman equation is used.

The Bellman equation is a fundamental recursive equation in dynamic programming and reinforcement learning, see for e.g., [13]. It helps to determine the optimal policy in decision-making problems. The Bellman equation is typically written as

$$V_\pi(s) = \max_a \left[\sum_{s'} P(s'|s,a) \left(R(s,a) + \gamma V_\pi(s') \right) \right], \quad (12)$$

where

- $V_\pi(s)$ is the value function, representing the expected cumulative reward from state s applying policy π.
- $R(s,a)$ is the immediate reward received after taking action a in state s.
- γ is the discount factor (between 0 and 1), which determines the importance of future rewards.
- $P(s'|s,a)$ is the transition probability of reaching state s' from s given action a.
- π is a plan or strategy that outlines the actions to be executed in every given state.

The idea of MDP is to chose a policy π^* such that

$$\pi^*(s) = \arg\max_a \left[\sum_{s'} P(s'|s,a) \left(R(s,a) + \gamma V_{\pi^*}(s') \right) \right]. \quad (13)$$

In others words, Bellman equation maximizes the expected value $V_\pi(s) = \mathbb{E}[G_n | \mathcal{X}_n = i]$, where $G_n := \sum_{k=0}^{\infty} \gamma^k R_{n+k+1}$.

Thus far, we have outlined the equations modeling MDPs, but these models have certain limitations. In particular, the sojourn time within a state is a random variable modeled by a geometric distribution (in the discrete case). Therefore, the next subsection will be introduced SMDPs, where the sojourn time can be represented by any distribution function.

4.4 semi-Markov Decision Process (SMDP)

MDP have some disadvantages respect of the distribution function of the sojourn time in a state, for this reason in this article we propouse a SMDP to successfully achieve the landing of a UAV on a moving platform.
It is well known that if process $\{\mathcal{X}_t\}$ is an SMC then the bivariate process $\{\mathcal{X}_t, B_t\}$ is a Markov chain, see, e.g., [17], where its transition probability matrix $\mathcal{P}$ is a function of the semi-Markov kernell as follows

$$\mathcal{P}(i,b_1)(j,b_2) = \begin{cases} \frac{q_{i,j}(b_1+1)}{\overline{H}_i(b_1)}, & \text{if } i \neq j \text{ and } b_2 = 0; \\ \frac{\overline{H}_i(b_1+1)}{\overline{H}_i(b_1)}, & \text{if } i = j \text{ and } b_2 - b_1 = 1. \end{cases} \quad (14)$$

where $\overline{H}_i(b) := \mathbb{P}(T_1 > b \mid \mathcal{Y}_0 = i) = 1 - H_i(b)$.

We use bivariate process $\{\mathcal{X}_t, B_t\}$ to select the best action according with the state and the backward of the system, in others words, we search a policy $\pi^{*'}$ such that

$$\pi^{*'}(s) = \arg\max_a \left[\sum_{s'} P(s'|s,b,a) \left(R(s,b,a) + \gamma V_{\pi^{*'}}(s') \right) \right]. \quad (15)$$

The main difference between Equiations (13) and (15) is that the transition probabilities and rewards now depend on the time the system has spent in the present state which is a random variable, modeled by any discrete time distribution function.

4.5 Implementation

A Semi-Markov Decision Process (SMDP) is employed to enable UAV landing on a moving platform. The system defines five discrete states and five possible actions, summarized in Table 1.

http://www.imavs.org/

Table 1: SMDP States and Actions

State	Description	Action	Description
$S1$	Platform not in sight	$A1$	Move in circles
$S2$	Platform in sight	$A2$	Center code
$S3$	Platform below	$A3$	Move forward
$S4$	Landed (fail)	$A4$	Move backward
$S5$	Landed (success)	$A5$	Land

Available actions are $\{A1, A3, A4, A5\}$ in $S1$, and $\{A1, A2, A3, A4, A5\}$ in $S2$ and $S3$. States $S4$ and $S5$ are absorbing; any action leads back to the same state with probability one.

Rewards guide the UAV's decisions by encouraging progress and penalizing undesired behavior. The reward function is empirically set as:

- Progress to next state: $+3$
- Remain in current state: $+1$
- Return to previous state: -3

In a SMC the semi-Markov kernel compuetes transition probabilities between states, see Equation (10). The empirical estimator for the semi-Markov kernel used in this article is given by following equation

$$\hat{K}_{ij}(t, M) := \frac{N_{ij}(t, M)}{N_i(M)} \quad \text{for } i, j \in S; t \in \mathbb{N}, \tag{16}$$

where $M \in \mathbb{N}^*$ is a sensor time and $N_{ij}(t, M)$ is the number of transitions on the embedded Markov chain $\{\mathcal{Y}_t\}$ from i to j up to time M, with sojourn time t such that $1 \leq t \leq M$. $N_{ij}(t, M)$ is formally defined by

$$\begin{aligned} N_{ij}(t, M) &:= \sum_{n=1}^{N(M)} \mathbf{1}_{\{\mathcal{Y}_{n-1}=i, \mathcal{Y}_n=j, T_n=t\}} \\ &= \sum_{n=1}^{M} \mathbf{1}_{\{\mathcal{Y}_{n-1}=i, \mathcal{Y}_n=j, T_n=t, \tau_n \leq M\}}. \end{aligned}$$

Therefore

$$N_i(M) := \sum_{t=0}^{\infty} \sum_{j \in S} N_{ij}(t, M)$$

Notice that $\mathbf{1}_{\{x=A\}}$ is the indicator function of event A, that means

$$\mathbf{1}_{\{x=A\}} = \begin{cases} 1 & if \;\; x = A; \\ 0 & \text{otherwise.} \end{cases}$$

The data for computing Equation(16) where taken from several flights.

4.6 Algorithm Implementation

The following steps outline the implementation of the mathematical framework described in Section 4:

1. For every $s \in S$ initialize $V_\pi(s)$, with $V_\pi(s) = 0$ for non-absorbing states.
2. Iteratively update $V_\pi(s)$ using the Bellman equation.
3. Compute the optimal policy $\pi^{*'}(s)$ based on $V_\pi(s)$.

5 EXPERIMENTAL VALIDATION

Experimental validation is a key component of this study. The UAV states, actions, and rewards are defined based on manual landing trials. A DJI Tello UAV is selected for its compact size, efficiency, and ease of programming.

In the first experimental stage, landings on a platform moving at constant linear velocity in one dimension are performed. The UAV's position is adjusted to match the platform's motion, while manual control is applied to simulate approach, alignment, descent, and touchdown phases. Each maneuver is associated with specific actions and rewards to reinforce desired behaviors.

The collected data is used to refine the SMDP model, enabling more robust decision-making for autonomous landing.

After the first stage in experimentation the following computations were obtained based on the provided transition data.

5.1 Transition Probability and Reward Matrices

Table 2 shows a sample of the derived transition probability (P) and rewards (R) of the embedded Markov chain $\{\mathcal{Y}_t\}$ of SMC $\{\mathcal{X}_t\}$ of the UAV.

Table 2: Transition Probability and Reward Matrices

State	Action	Next State (s')	$P(s'\|s, a)$	$R(s, a)$
1	1	2	0.0	1.0
1	1	3	0.0	1.0
2	2	3	1.0	3.0
3	3	4	1.0	-3.0

5.2 Value Function and Optimal Policy

The value function $V_\pi(s)$ after convergence and the optimal policy $\pi^*(s)$ are shown in Table 3. This table helps the UAV understand the best decision based on its actual state.

Table 3: Value Function $V_\pi(s)$ and Optimal Policy $\pi^*(s)$

State (s)	Value ($V(s)$)	Optimal Action (a)
1	12.76	1
2	11.76	5
3	9.99	3
4	0.0	–
5	0.0	–

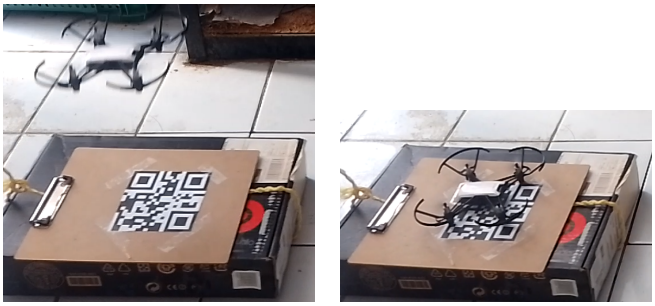
http://www.imavs.org/

5.3 Results

The experimental results demonstrate a progressive improvement in the UAV's landing performance as the SMDP model undergoes training and parameter adjustments. Initial landing attempts showed a high failure rate, highlighting the need for fine-tuning both the decision-making framework and control parameters. However, as the system iteratively learned from previous experiences, the success rate significantly increased, reflecting the model's ability to adapt and optimize its performance over time.

By the end of the experimental phase, the UAV consistently achieved successful landings with greater precision and stability. To further illustrate the system's evolution and final performance, a demonstration video is provided, showcasing representative landing sequences throughout the training process:
`https://youtu.be/FTALlxdZ64E`.

Some screenshots have been taken from the video and depicted in Figures 3a and 3b



(a) Drone approaching (b) Drone landing

Figure 3: Drone landing procedure

Following figures represent the results of a series of flights. The first graph represents a comparative analysis of the percentage of successful drone landings over five-flight intervals using three different control strategies: manual landing (Non-Markov), Markov, and Semi-Markov processes.

The experiment involved grouping 25 drone flights into five segments of five flights each and calculating the success rate for each method in every interval. The Non-Markov approach, which lacks probabilistic decision-making, showed lower and inconsistent success rates. The Markov approach, which uses memoryless transitions between states, performed better but had fluctuations.

The Semi-Markov approach, which considers both state transitions and time delays, achieved the highest and most stable success rates as depicted in Figure 4.

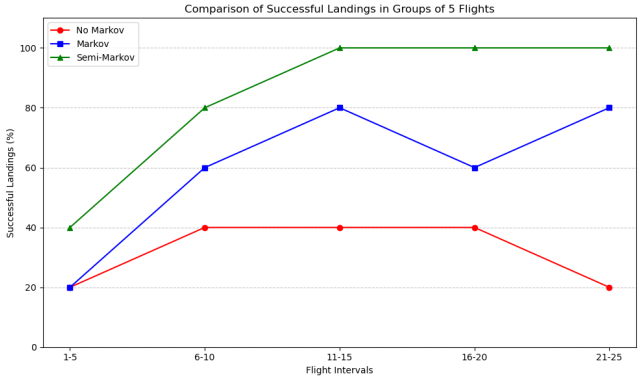


Figure 4: Comparison of successful landings

The effectiveness of three different landing strategies (Non-Markov, Markov, and Semi-Markov) is evaluated by measuring the closest distance to the platform across five flight trials, see Figure 5. Each strategy was applied to a drone during landing attempts, and for each flight, the shortest distance achieved from the target at the final stage of the flight was recorded. The improvement in landing accuracy over successive flights is represented in the graph, with the Y-axis indicating the closest distance to the platform (0–30 cm, where lower is better) and the X-axis showing the flight number. It is observed that the most precise landings are consistently achieved with the Semi-Markov strategy, followed by the Markov strategy, while the Non-Markov approach shows the least improvement over time. These results suggest that the drone's landing accuracy is enhanced over repeated attempts when probabilistic decision-making models, particularly Semi-Markov processes, are incorporated.

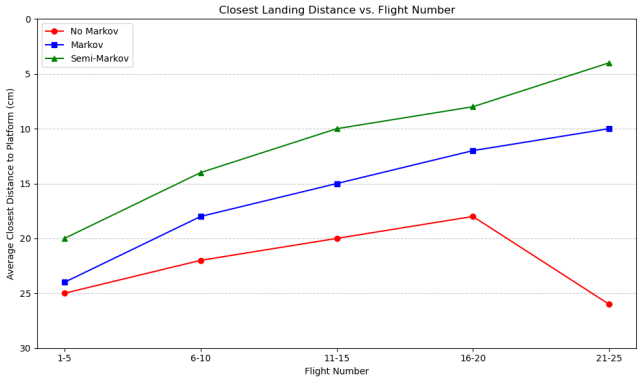


Figure 5: Comparison of distance from target

6 Conclusion

This paper demonstrates the effectiveness of Markov Decision Processes (MDP) and Semi-Markov Decision Processes (SMDP) for UAV autonomous landing. These models provide robust tools for real-time decision-making under

uncertainty, as validated by experimental results showing improved performance over non-probabilistic approaches.

The system exhibited notable gains in robustness and decision accuracy. Initially, high failure rates highlighted the challenges of dynamic environments, but iterative learning led to progressively higher landing success, reflecting the adaptability of the SMDP framework.

A key advantage of SMDPs was their capacity to model temporal dynamics, outperforming standard MDPs by accounting for state transitions and time delays. This resulted in greater landing precision and stability.

Future work will focus on mitigating the system's tendency to remain in certain states too long, improving responsiveness via enhanced temporal modeling and adaptive learning. Testing in more complex, unpredictable scenarios will further assess scalability and robustness.

Probabilistic models (particularly SMDPs) prove valuable for improving UAV autonomy, offering a path toward more reliable and efficient real-world deployment.

http://www.imavs.org/

ACKNOWLEDGEMENTS

The authors would like to express their gratitude to the Computer Science Department at Tecnológico de Monterrey, Guadalajara Campus, for providing the resources and institutional support that made this research possible.

REFERENCES

[1] Alvika Gautam, PB Sujit, and Srikanth Saripalli. A survey of autonomous landing techniques for uavs. In *2014 international conference on unmanned aircraft systems (ICUAS)*, pages 1210–1218. IEEE, 2014.

[2] Xiaoshan Bai, Youqiang Ye, Bo Zhang, and Shuzhi Sam Ge. Efficient package delivery task assignment for truck and high capacity drone. *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[3] Mayank Mittal, Rohit Mohan, Wolfram Burgard, and Abhinav Valada. Vision-based autonomous uav navigation and landing for urban search and rescue. In *The International Symposium of Robotics Research*, pages 575–592. Springer, 2019.

[4] Jonathan Sprinkle, J Mikael Eklund, and S Shankar Sastry. Deciding to land a uav safely in real time. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 3506–3511. IEEE, 2005.

[5] François Chaumette. Visual servoing. In *Computer Vision: A Reference Guide*, pages 1367–1374. Springer, 2021.

[6] Francois Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. In *The confluence of vision and control*, pages 66–78. Springer, 2007.

[7] Efstratios Kakaletsis, Charalampos Symeonidis, Maria Tzelepi, Ioannis Mademlis, Anastasios Tefas, Nikos Nikolaidis, and Ioannis Pitas. Computer vision for autonomous uav flight safety: An overview and a vision-based safe landing pipeline example. *Acm Computing Surveys (Csur)*, 54(9):1–37, 2021.

[8] 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.

[9] Florent Le Bras, Tarek Hamel, Robert Mahony, Christian Barat, and Julien Thadasack. Approach maneuvers for autonomous landing using visual servo control. *IEEE transactions on aerospace and electronic systems*, 50(2):1051–1065, 2014.

[10] 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.

[11] Alessandra Elisa Sindi Morando, M Ferreira Santos, P Castillo, and A Correa-Victorino. Vision-based algorithm for autonomous aerial landing. In *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 652–657. IEEE, 2024.

[12] Martin L Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

[13] Richard S Sutton. Reinforcement learning: An introduction. *A Bradford Book*, 2018.

[14] Riccardo Polvara, Massimiliano Patacchiola, Sanjay Sharma, Jian Wan, Andrew Manning, Robert Sutton, and Angelo Cangelosi. Toward end-to-end control for uav autonomous landing via deep reinforcement learning. In *2018 International conference on unmanned aircraft systems (ICUAS)*, pages 115–123. IEEE, 2018.

[15] Jacques Janssen and Nikolaos Limnios. *Semi-Markov models and applications*. Springer Science & Business Media, 2013.

[16] Odile Bourquardez, Robert Mahony, Nicolas Guenard, François Chaumette, Tarek Hamel, and Laurent Eck. Image-based visual servo control of the translation kinematics of a quadrotor aerial vehicle. *IEEE Transactions on Robotics*, 25(3):743–749, 2009.

[17] Vlad Stefan Barbu and Nikolaos Limnios. *Semi-Markov chains and hidden semi-Markov models toward applications: their use in reliability and DNA analysis*, volume 191. Springer Science & Business Media, 2009.

Model-Based Analysis of UAV Accurate Landing in Stochastic Turbulent Environments

Zhanghao Lu*, Yuan Tang, Lutong Li, Renbo Su, Zhengyi Jiang, Simon Watson†, and Andrew Weightman‡
Manchester Centre for Robotics and AI, University of Manchester, Oxford road, Manchester

http://www.imavs.org/

Abstract

Deploying Inspection, Maintenance, and Repair (IMR) robots using unmanned aerial vehicles (UAVs) has emerged as a promising method for wind turbine blade inspection due to lower costs and higher accuracy. However, deploying these robots remains a challenge due to the high-speed, turbulent wind environments around wind farms, making it difficult for UAVs to land on turbine blades accurately. Using a novel discrete stochastic turbulent wind field model, this study evaluates a landing strategy for UAVs to land at higher speeds compared to typical UAV landing speed ($\leq$ 0.5 m/s) in turbulent wind fields to minimise lateral landing errors. Based on 4800 simulations performed at various wind speeds (3.0 - 12.0 m/s) and UAV landing speeds (0.5 - 3.0 m/s), it was observed that this landing strategy provides lower mean landing errors (up to 37.7%) compared to existing UAV landing systems when the wind speed is below 9 m/s, improving landing success rates by up to 30%.

1 Introduction

In recent decades, offshore wind farms have gained momentum due to fewer land-use constraints and access to stronger, more consistent winds compared to onshore wind farms [1]. However, the harsh marine environment has increased the demand for blade inspection, maintenance and repair (IMR), with blade damage being the most frequent and costly [2].

Traditional manual methods for offshore inspection are risky and expensive due to the unique geographical location [3]. Although UAVs offer promising solutions for remote blade inspection [4], they still face key limitations such as flight instability in turbulence, limited inspection accuracy, and battery capacity constraints. Moreover, maintenance and repair still require manual intervention, limiting full automation and cost reduction.

To address these challenges, a multi-robot platform was proposed in [3], where UAVs deploy and retrieve IMR robots for close-contact blade tasks. Among the mission phases, the landing poses the highest risk due to direct contact between the UAV and the blade under potentially high-speed turbulent environments.

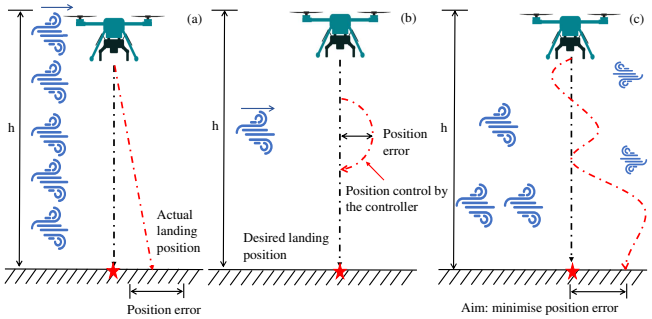


Figure 1: The illustration of the impact of (a) constant wind, (b) wind gusts and (c) wind turbulence on the positional error of the UAV.

The risks primarily arise from two factors: (1) lateral displacement caused by gusts and turbulence, which can reduce landing accuracy or lead to a landing near the blade edge, potentially causing the UAV to fall or collide with surrounding structures; and (2) vertical shear that might increase descent speed, resulting in elevated impact energy and contact forces, posing a risk of damaging both the UAV and the blade. While most existing studies adopt low descent speeds (typically $\leq$ 0.5 m/s), which allow sufficient time to compensate for gust-induced deviations [5], such approaches may be inadequate in highly turbulent conditions where wind speed and direction vary rapidly. Therefore, this study proposes a hypothesis that increasing descent speed can reduce the UAV's exposure time to wind disturbances and thereby improve landing accuracy, if the kinetic energy can be adequately dissipated during the landing process. To validate this hypothesis, we developed a simulation framework with a UAV platform and a height-varying stochastic turbulence field to evaluate landing performance across multiple wind and descent speed scenarios.

1.1 Background and Related Work

As illustrated in Figure 1, wind disturbances affecting UAV landing can be broadly classified into constant wind, gusts, and turbulence [6]. Constant wind refers to steady airflow that causes persistent deviations from the intended path, while gusts are short-term variations in speed and direction that lead to temporary trajectory shifts. Turbulence, the most

*Email address: zhanghao.lu@student.manchester.ac.uk
†Email address: simon.watson@manchester.ac.uk
‡Email address: andrew.weightman@manchester.ac.uk

http://www.imavs.org/

complex type, involves chaotic multidirectional fluctuations and is also observed in offshore environments [7]. Due to the time-varying nature of both wind magnitude and direction in turbulent conditions, UAVs may exhibit highly diverse and unpredictable landing trajectories.

Although most commercial UAVs report a wind resistance rating of up to 10 m/s [8], perfectly constant wind conditions are rarely observed in real-world environments [6]. Hence, most existing studies focus on reducing trajectory deviations under gust scenarios. Depending on the robustness of the controller and its compensation mechanisms, UAVs may recover their trajectories after such disturbances [9]. Most control approaches are based on Proportional-Integral-Derivative (PID) or Model Predictive Control (MPC) frameworks, enhanced with additional feedback, constraints, or compensation terms. These methods have been validated experimentally using high-power fans or through simulations with step-like gusts. More recently, reinforcement learning-based control strategies have also been explored and shown to be effective for mitigating gust-induced errors [10, 11].

However, simplified gust models with fixed direction and magnitude are insufficient to capture the complexity of the highly dynamic wind environment observed near offshore wind farms. For instance, the model in [11] introduces a linear ramp with Gaussian noise to simulate variability, but is limited to the horizontal plane and neglects vertical wind effects. In contrast, turbulent wind models incorporating time-varying fluctuations across three dimensions offer a more realistic representation of offshore wind environments.

Turbulent wind fields can be accurately simulated using computational fluid dynamics (CFD) [12], but the high computational cost makes CFD unsuitable for large-scale simulations required to assess the influence of descent speed and wind intensity on UAV landing performance.

To address this, many UAV studies adopt stochastic turbulence models based on power spectral density (PSD), such as the Dryden and von Kármán models [13, 14]. These models, grounded in stochastic theory and empirical data, provide a balance between accuracy and efficiency [6].

1.2 Summary of the Related Work

Section 1.1 reviews various wind field modelling approaches, as well as control and landing strategies to reduce landing errors. Table 1 summarises the key elements of the discussed UAV landing systems.

Most existing research on UAV landing strategies has concentrated on landing controllers for lightweight platforms ($\leq$2.5 kg) operating under unidirectional gust conditions, leaving three key gaps: (1) the influence of multidimensional variability in wind speed and direction on UAV landing accuracy has been largely overlooked; (2) the applicability of these strategies to high-payload UAVs remains uncertain; and (3) most importantly, except for the study by Olaz et al. [11], which specified a descent speed below 0.8 m/s, the role of descent speed in landing accuracy has been substantially underexplored, despite its direct impact on the UAV's exposure time to disturbances [5].

To address these gaps, this study develops integrated models of the UAV system and the three-dimensional turbulent wind field to provide a novel investigation into the effect of descent speed on the landing accuracy of high-payload UAVs under stochastic turbulent wind conditions.

2 Design Specifications for UAV Safe and Accurate Landing

Deploying IMR robots onto wind turbine blades using UAVs in turbulent environments requires the systematic quantification of key parameters related to the UAV platform, blade characteristics, environmental conditions, and the IMR robot. Currently, there is a lack of standardised design requirements for UAV landings on wind turbine blades under turbulent conditions. To ensure that the model aligns more closely with real-world offshore environments and application scenarios, this study proposes a set of design requirements for achieving safe and accurate UAV landings on wind turbine blades.

2.1 UAV platform and IMR robot specifications

A custom-built UAV platform named Goliath was developed in [3] for deploying a 4.5 kg IMR robot. Its capability to land on wind turbine blades was validated through field experiments under wind speeds of approximately 7.2 m/s and gusts reaching 9.2 m/s. The UAV was built upon a 3DXR-IND1000 octocopter frame with overall dimensions of 800 mm in width and depth and 500 mm in height. The UAV has an approximate mass of 14 kg. It was powered by eight X-U8II-KV85 motors equipped with MF2815 propellers, controlled via ALPHA 60A HV electronic speed controllers (ESCs), providing sufficient thrust to support a maximum take-off mass (MTOM) of approximately 30 kg.

2.2 Design requirements for Accurate and Safe Landing on Wind Turbine Blade Surface

In this study, wind turbine blades were designated as landing targets. Several design requirements were made based on a combination of existing literature, relevant regulations and standards, and field data collected from real-world offshore environments.

To ensure safe operation, the UAV must maintain a minimum stand-off distance of 5 m from the blade during inspection to prevent collision [19]. Therefore, the landing height range in this study was defined as a 5 m interval from the completion of the inspection to the landing on the blade surface. Furthermore, during the landing process, the UAV landing system should be capable of maintaining its attitude within 30 degrees even under variations in wind magnitude and direction [20].

A wind turbine blade can be divided into three sections: the root, the mid-section, and the tip [21]. While the root sec-

Table 1: Comparison of UAV landing systems in different wind conditions.

Study	Methodology	Wind type	Max. wind speed (m/s)	Landing error (mm)	System mass (kg)	Validation
Todeschini et al. [15]	MPC + hierarchical feedback controller	Constant wind (3D)	3.0	N/A	11.0	Simulation
Ghadiok et Al. [16]	Extensive filtering & altitude estimation	Gust (1D)	2.8	<500	1.4	Fan
Alexis et al. [14]	MPC + Piecewise Affine	Gust (3D)	4.3	N/A	2.5	Fan
Hentzen et al. [9]	MPC	Gust (1D)	12.0	<150.0	0.7	Fan
Mendez et al. [17]	Wind-preview-based MPC	Gust (1D)	10.0	<200.0	2.4	Simulation
Saj et al. [10]	RL	Gust (1D)	5.0	<50	1.4	Fan
Olaz et al [11]	RL	Gust (2D)	10.0	<500.0	N/A	Simulation
Sydney et al. [18]	Feedback linearisation controller	Turbulence (3D)	2.0	<400.0	N/A	Simulation

tion has the longest chord length, its high curvature makes it unsuitable for UAV landing due to the increased risk of flipping over. In contrast, the mid-section and tip have shorter chord lengths (approximately 2.7 m, based on the DTU 10-MW reference wind turbine [22]) but offer flatter surfaces.

To further enhance landing safety, operations near the blade edges should be avoided. A 0.2 m buffer zone was reserved on each side to reduce the risk of UAV fall-off, resulting in a defined landing zone width of 1.8 m, with 0.9 m on either side of the blade's central axis, as shown in Figure 2.

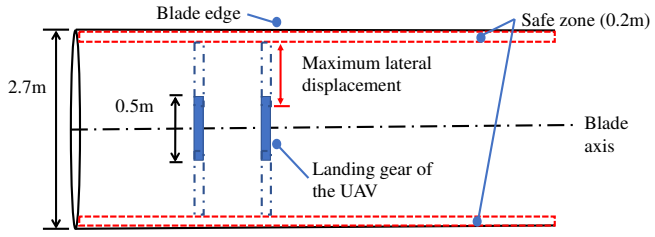


Figure 2: The illustration of the appropriate landing zone on the wind turbine blade surface and the maximum lateral displacement for the UAV to land accurately and safely on a wind turbine blade.

Offshore wind farms are subject to highly variable wind conditions, resulting in unpredictable wind speeds. To characterise typical conditions, a dataset of maximum wind speeds recorded from March to September 2013 at the Methil offshore wind farm, UK was analysed [3]. The results showed that wind speeds followed a normal distribution (7.82 ± 3.89 m/s). Wind speeds below 12 m/s account for 86.2% of all observations, supporting the selection of 12 m/s as the maximum threshold in this study.

3 UAV LANDING STRATEGY AND PLATFORM MODEL

3.1 UAV Landing strategy

Before modelling the UAV platform and the wind field, the UAV landing strategy must first be established. We assume that once the UAV carrying an IMR robot moves above the wind turbine blade, it has already identified a suitable landing spot during the scanning phase and hovers above it. After stabilising its attitude, the UAV accelerates from 0 m/s to the desired speed and then descends to the blade surface at this controlled speed through the wind field.

3.2 UAV Platform Model

The UAV platform model is based on the concepts presented in [20] and consists of four modules: kinematic, dynamic, control effectiveness, and propulsor models.

The UAV model is simplified as a six-degree-of-freedom (6-DOF) rigid body with four motors mounted in a cross configuration. Two right-handed coordinate frames of the Aircraft-Body Coordinate Frame (ABCF) and the Earth-Fixed Coordinate Frame (EFCF) are defined and denoted by the subscripts *b* and *e*.

3.2.1 UAV Kinematic Model

The position vector of the UAV's centre of mass (COM) is defined in the Earth-Fixed Coordinate Frame (EFCF) $\mathbf{p}_e = [x_e, y_e, z_e]^T \in \mathbb{R}^3$ with units of m, and its time derivative $\dot{\mathbf{p}}_e = \mathbf{v}_e$ represents the UAV's velocity in m/s.

The attitude of the UAV is represented using quaternions to avoid the singularity problem associated with Euler angle representations. The rotation from EFCF to ABCF is represented by $\mathbf{q}_e^b \triangleq [q_0\, q_1\, q_2\, q_3]^T \triangleq [q_0\, \mathbf{q}_V^T]^T$ (dimensionless, with $|\mathbf{q}_e^b| = 1$). The relationship between the quaternions and angular velocities of the UAV body can then be expressed as

$$\dot{\mathbf{q}}_e^b = \begin{bmatrix} \dot{q}_0 \\ \dot{\mathbf{q}}_V \end{bmatrix} = \frac{1}{2} \begin{bmatrix} -\mathbf{q}_V^T \\ q_0 \mathbf{I}_3 + [\mathbf{q}_V]_\times \end{bmatrix} \boldsymbol{\omega}_b, \tag{1}$$

where $\omega_b = [\omega_x\, \omega_y\, \omega_z]^T$ represents the angular velocity vector of the UAV body in rad/s, $\mathbf{I}_3$ is an identity matrix (dimensionless), and $[\mathbf{q}_V]_\times$ is the skew-symmetric form of $\mathbf{q}_V$ (dimensionless).

3.2.2 UAV Dynamic Model

The UAV dynamic model consists of translational and rotational representations, which are derived using the Newton-Euler equations [20]:

$$\dot{\mathbf{V}}_e = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \frac{1}{m} R_b^e \begin{bmatrix} 0 \\ 0 \\ f_z \end{bmatrix} + \frac{1}{m} \begin{bmatrix} -k_x \dot{x}_e \\ -k_y \dot{y}_e \\ -k_z \dot{z}_e \end{bmatrix}, \tag{2}$$

http://www.imavs.org/

http://www.imavs.org/

$$J \cdot \dot{\omega}_b = -\omega_b \times (J \cdot \omega_b) + \boldsymbol{\tau}, \tag{3}$$

where g is the gravitational acceleration in m/s^2, m is the UAV mass in kg and f_z denotes the total thrust generated by the propellers in N. The coefficients k_x, k_y, k_z are dimensionless drag coefficients along the respective axes. $J = \text{diag}(J_{xx}, J_{yy}, J_{zz}) \in \mathbb{R}^{3\times3}$ is the UAV's moment of inertia matrix with units of $kg \cdot m^2$, and the vector $\boldsymbol{\tau} \triangleq [\boldsymbol{\tau}_x \, \boldsymbol{\tau}_y \, \boldsymbol{\tau}_z] \in \mathbb{R}^3$ represents the body-frame torques generated by the propellers in $N \cdot m$.

The rotation matrix R_b^e from ABCF to EFCF was obtained from $\mathbf{q}_e^b$ using the standard conversion of quaternion to DCM (Direction Cosine Matrix). Both R_b^e and $\mathbf{q}_e^b$ are dimensionless. For simplicity, the gyroscopic torques generated by the spinning motors were neglected in this model.

3.2.3 Control effectiveness and propulsor model

The propeller thrust and dynamic reaction torque produced by each motor were modelled as:

$$T_i = C_T \varpi_i^2, \tag{4}$$

$$M_i = C_M \varpi_i^2 + J_{RP} \dot{\varpi}_i, \tag{5}$$

where C_T and C_M are the dimensionless coefficients of thrust and torque. The total thrust and torques of the UAV are then determined by the angular velocities of the propellers ϖ_i in rad/s:

$$\begin{bmatrix} f \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} C_T & C_T & C_T & C_T \\ \frac{\sqrt{2}dC_T}{2} & \frac{-\sqrt{2}dC_T}{2} & \frac{-\sqrt{2}dC_T}{2} & \frac{\sqrt{2}dC_T}{2} \\ \frac{\sqrt{2}dC_T}{2} & \frac{\sqrt{2}dC_T}{2} & \frac{-\sqrt{2}dC_T}{2} & \frac{-\sqrt{2}dC_T}{2} \\ C_M & -C_M & C_M & -C_M \end{bmatrix} \begin{bmatrix} \varpi_1^2 \\ \varpi_2^2 \\ \varpi_3^2 \\ \varpi_4^2 \end{bmatrix}, \tag{6}$$

where d denotes the distance from each motor to the UAV's COM in m.

In turbulent environments, accurately modelling the motor's dynamic response is essential for capturing the UAV's real-time behaviour and landing performance. Therefore, this study incorporates both the motor and the ESC in the propulsor model. The ESC outputs a pulse width modulation (PWM) signal, which is converted to a dimensionless throttle command $\sigma \in [0, 1]$. At steady state, the motor speed ϖ_{ss} (in rad/s) is assumed to vary linearly with the throttle input:

$$\varpi_{ss}(\sigma) = C_R \sigma + \varpi_b,$$

where C_R and ϖ_b are the dimensionless gain and offset, respectively. To model the motor's response delay, a first-order dynamic response function with a time constant T_m is used, resulting in:

$$\varpi(s) = \frac{1}{T_m s + 1}(C_R \sigma + \varpi_b), \tag{7}$$

where T_m denotes the time required for the motor speed to reach 63.2% of its steady-state value (in s).

3.2.4 UAV control system

To evaluate the impact of descent velocity on UAV landing performance, a classic cascaded PID control system is adopted, commonly used in commercial UAVs [23]. The control system comprises four modules: position controller, attitude controller, motor controller, and the UAV platform model. The position controller includes vertical and horizontal sub-controllers.

The quadrotor UAV is an underactuated system with four control inputs—total thrust $f \in \mathbb{R}$ and body-axis torques $\boldsymbol{\tau} \in \mathbb{R}^3$—but six outputs: position $\mathbf{p} \in \mathbb{R}^3$ and attitude $\boldsymbol{\Theta} \in \mathbb{R}^3$. Consequently, only the desired position $\mathbf{p}_d$ and yaw angle Ψ_d are directly tracked, while the desired roll ϕ_d and pitch θ_d are computed from the horizontal position controller using the desired yaw angle and desired acceleration along the x and y axes.

Thrust and attitude commands are calculated by the PID controller and mapped to motor throttle inputs. To preserve attitude control margin, the thrust output was capped at 90% of the UAV's maximum capacity.

4 STOCHASTIC WIND TURBULENCE MODEL

As mentioned in Section 1.1, PSD-based methods are more suitable for simulating turbulent offshore environments for UAV landing scenarios. Compared to the Dryden model, the Von Kármán model shows better agreement with continuous turbulence measurements [24]. In this study, a stochastic wind turbulence model is constructed based on the Von Kármán model provided in the MATLAB/Simulink Aerospace Blockset (MIL-F-8785C).

4.1 Von Kármán Wind Turbulence Model

The von Kármán wind turbulence model generates time-varying wind velocities by applying frequency-domain turbulence filters to white noise inputs. These filters, defined as functions of the Laplace variable s, shape the spectral characteristics of wind disturbances in the longitudinal, lateral, and vertical directions. The corresponding transfer functions for each direction are given as follows:

$$\Phi_{V_{wind_u}}(s) = \frac{2\sigma_u^2 L_u}{\pi V} \cdot \frac{1}{\left[1 + \left(1.339 \frac{L_u}{V} s\right)^2\right]^{5/6}} \tag{8}$$

$$\Phi_{V_{wind_v}}(s) = \frac{\sigma_v^2 L_v}{\pi V} \cdot \frac{1 + \frac{8}{3}\left(1.339 \frac{L_v}{V} s\right)^2}{\left[1 + \left(1.339 \frac{L_v}{V} s\right)^2\right]^{11/6}} \tag{9}$$

$$\Phi_{V_{wind_w}}(s) = \frac{\sigma_w^2 L_w}{\pi V} \cdot \frac{1 + \frac{8}{3}\left(1.339 \frac{L_w}{V} s\right)^2}{\left[1 + \left(1.339 \frac{L_w}{V} s\right)^2\right]^{11/6}}, \tag{10}$$

http://www.imavs.org/

where $\Phi_{V_{wind_u}}$, $\Phi_{V_{wind_v}}$, and $\Phi_{V_{wind_w}}$ are the power spectral densities of wind velocity in the x, y, and z directions, respectively. V represents the relative velocity between the UAV and the ambient wind. The parameters σ_u, σ_v, and σ_w indicate the turbulence intensities, while L_u, L_v, and L_w correspond to the turbulence scale lengths in each respective direction. These transfer functions are applied to white noise signals generated by a random source, producing fluctuations in external wind speed that characterise the temporal evolution of turbulence [25]. Different noise seeds yield distinct stochastic wind profiles.

According to the military specification MIL-F-8785C [26], the turbulence intensity σ and scale length L can be computed as functions of flight altitude h using the following:

$$\begin{aligned} L_w &= h, \\ L_u = L_v &= \frac{h}{(0.177 + 0.000823h)^{1.2}}, \end{aligned} \tag{11}$$

$$\begin{aligned} \sigma_w &= 0.1W_{20}, \\ \frac{\sigma_u}{\sigma_w} = \frac{\sigma_v}{\sigma_w} &= \frac{1}{(0.177 + 0.000823h)^{0.4}}, \end{aligned} \tag{12}$$

where h is the flight altitude of the UAV and W_{20} defines the average wind speed at 20 feet (6 m).

4.2 *Discrete Turbulent Wind Field Varying with Altitude*

In a classical von Kármán turbulence model, both mean wind speed and turbulence intensity are assumed constant. However, real wind farm environments exhibit vertical wind gradients, leading to varying wind speeds along the blade height, typically within 1 m/s per metre [27].

To capture this effect, the vertical space between the blade and the UAV is discretised into five uniform height intervals. Each interval is assigned a unique W_{20} to reflect local wind intensity. The wind speed at the mid-height is defined as V_{mid}, with random perturbations introduced via two sequences $\Theta_1 = [x_1, x_2, \ldots, x_{\frac{n}{2}}]$ and $\Theta_2 = [y_1, y_2, \ldots, y_{\frac{n}{2}}]$, where $0 < x_i < 1$ and $-1 < y_i < 0$. Wind speeds are increased above and decreased below the mid-level by sequentially adding these values, yielding a smoothly varying vertical wind profile, as illustrated in Figure 3.

5 Simulation Setup and Methodology Summary

To evaluate whether higher descent speeds improve landing accuracy under turbulence, the UAV's descent velocity was varied from 0.5 m/s (a typical UAV landing speed [5]) to 3.0 m/s, the maximum descent speed limited by most commercial UAVs. Six discrete values: [0.5, 1.0, 1.5, 2.0, 2.5, 3.0] m/s were selected.

According to the design requirements in Section 2.2, the maximum wind speed is limited to 12 m/s. Since most UAVs

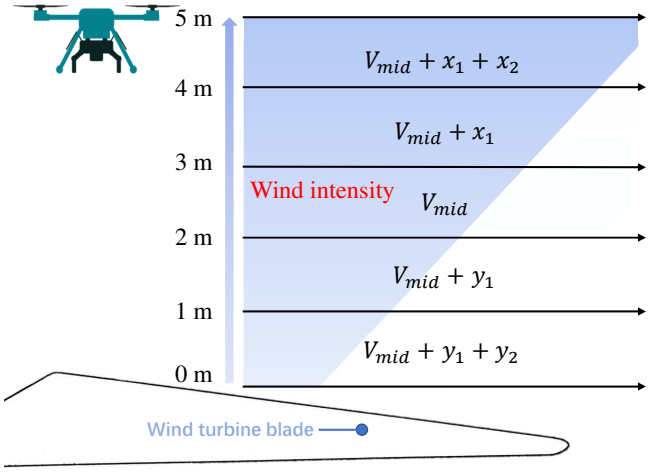


Figure 3: The illustration of the vertical wind gradients above the wind turbine blade.

can maintain landing accuracy under light breeze conditions (up to 3 m/s) [25], four representative values as [3.0, 6.0, 9.0, 12.0] m/s were selected for simulation.

Given the stochastic nature of the wind model, each simulation run generated a unique wind profile based on different random noise seeds. These seeds, along with the parameters Θ_1 and Θ_2, were recorded to ensure repeatability. For each wind speed setting, the same set of noise seeds was used across all descent speeds to enable a fair comparison.

Each combination of descent speed and wind speed was simulated 200 times, yielding a total of 4800 trials for statistical analysis. The choice of 200 trials per setting balances computational cost and statistical robustness, and is sufficient to capture the variability in landing outcomes caused by the stochastic wind conditions.

Simulations were conducted in MATLAB/Simulink using the ODE4 solver with a fixed time step of 0.001 s.

To provide a clearer overview of the simulation process, the integrated framework of the methodology is illustrated in Figure 4.

6 Simulation Results and Discussion

In order to evaluate the UAV's landing accuracy and safety across varying wind speeds and descent velocities, this section considers the distribution of descent trajectories, the final touchdown locations, and the overall landing success rates under different scenarios.

6.1 *Landing trajectories*

In the process of UAV landing, maintaining a stable vertical descent trajectory is essential to avoid collisions with turbine structures. Figure 5 presents representative 3D landing trajectories for descent speeds of 0.5 m/s and 3.0 m/s under turbulent wind conditions with a maximum wind speed of 6 m/s.

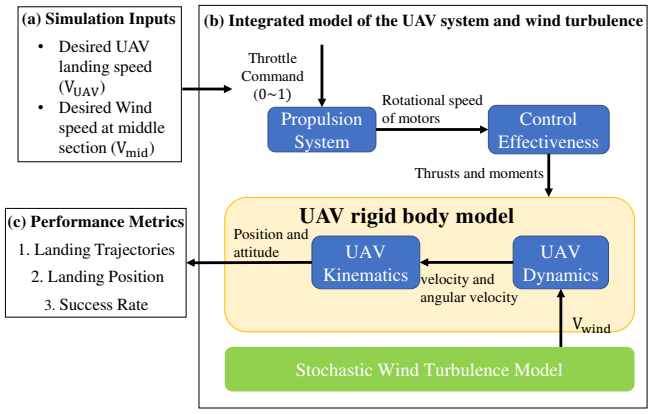


Figure 4: The summary diagram of the overall simulation flow, including (a) the simulation inputs; (b) the integrated model of the UAV system and the stochastic wind turbulence, where the wind model is embedded into the UAV dynamics; and (c) the performance metrics used to evaluate landing performance.

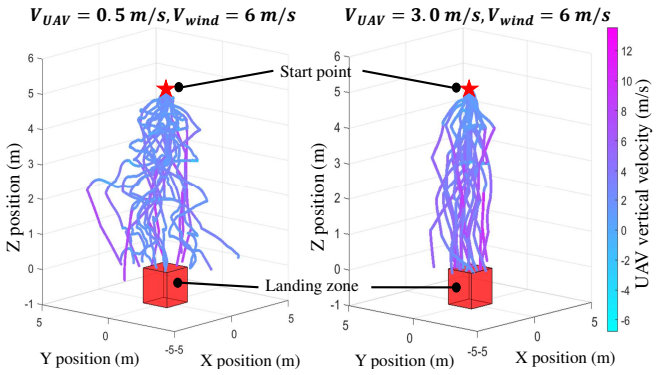


Figure 5: 20 samples of landing trajectories of UAV landing in 3D space when maximum wind speed is 6 m/s and a desired landing speed of 0.5 m/s and 3 m/s. The red box indicates the designated landing zone. The trajectories at 3.0 m/s are visibly more concentrated around the vertical descent path, while those at 0.5 m/s show greater lateral deviation. This suggests that descending at a higher speed in stochastic turbulent environments helps reduce the risk of collision with turbine structures.

The results show that despite the inherent randomness of the wind field, which could cause significant fluctuations in individual trajectories, a consistent overall descent pattern remained observable. A slower descent speed (0.5 m/s) leads to greater trajectory deviations, whereas a faster speed (3.0 m/s) results in trajectories more tightly concentrated around the vertical path. This also results in lower accuracy for the 0.5 m/s group, compared to the 3.0 m/s group, in terms of successfully reaching the designated landing zone.

The trajectories are colour-mapped according to the UAV's instantaneous descent speed, with visible colour fluctuations along some paths. These variations indicate disturbances in vertical motion caused by wind forces along the z-axis, which impair the UAV's ability to maintain its target descent speed.

6.2 *Landing position evaluation*

Figure 6 illustrates the landing positions of UAVs descending at speeds of 0.5 and 3.0 m/s under turbulent wind fields with maximum wind speeds of 3, 6, 9, and 12 m/s. Each marker represents the UAV's touchdown location on the turbine blade in a single simulation, while the red circle denotes the designated landing zone. The colour gradient of each point indicates its lateral displacement from the target, with lighter shades signifying a greater displacement.

All landing points exhibit a uniform radial distribution from the centre outward, suggesting the stochastic turbulent field affects the UAV symmetrically in all directions.

The results reveal that as wind speed increases, landing deviations become more pronounced, leading to more landings outside the designated target zone. However, across all wind conditions, UAVs descending at 0.5 m/s show a wider spread of landing positions and more outliers deviating from the central region. In contrast, at a descent speed of 3.0 m/s, the landing points are more concentrated near the centre. This improvement is particularly evident under wind speeds of 3.0 m/s and 6.0 m/s, with the most significant effect observed at 6.0 m/s, where the mean, median and standard deviations are reduced by 37.7%, 39.6% and 40.9%, respectively. However, under higher wind conditions (9.0 m/s and 12.0 m/s), the improvement diminishes, with the smallest gains observed at 12.0 m/s with reductions of 16.9%, 9.3% and 20.3%.

6.3 *Landing success rate*

Considering the safe landing zone defined in Section 2.2, Figure 6e presents a heat map illustrating the landing success rates across all tested scenarios. In every case, increasing the descent speed results in an improved success rate. Under relatively calm wind conditions with a maximum wind speed of 3 m/s, the UAV consistently maintained a success rate above 85%, reaching as high as 97.0% at a descent speed of 3.0 m/s—the highest success rate observed among all conditions. In contrast, within a 6 m/s wind field, the success rate only exceeded 50% when the descent speed was greater than 2.0 m/s; all other configurations failed to meet this threshold. However, the baseline success rate at 3 m/s wind speed was already high, resulting in relatively modest gains. Conversely, in the 6 m/s wind condition, the success rate improved significantly by approximately 30% as descent speed increased from 0.5 m/s to 3.0 m/s. This suggests that under moderately disturbed wind conditions, where UAV control is not completely compromised, increasing the descent speed can effectively enhance the probability of a successful and safe landing if the additional kinetic energy can be dissipated and

http://www.imavs.org/

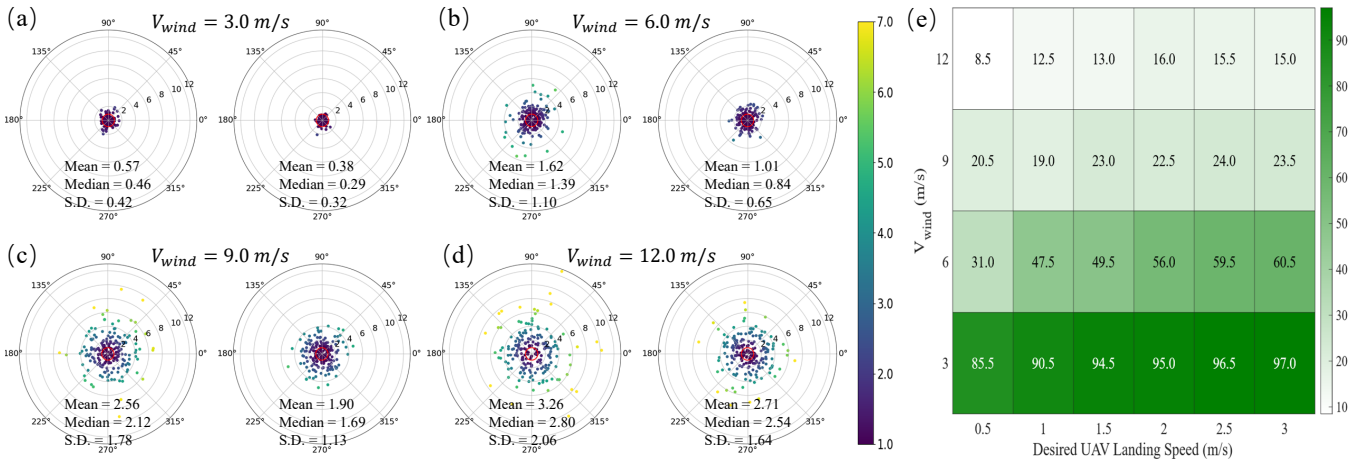


Figure 6: The bullseye graph displays the 200 samples of landing position of a UAV descending at 0.5 m/s and 3 m/s speeds in wind turbulence conditions of (a) 3 m/s, (b) 6 m/s, (c) 9 m/s, and (d) 12 m/s.

http://www.imavs.org/

not transferred to the wind turbine blades.

7 Conclusion and Future Work

In this study, we proposed a landing strategy that employs higher-than-typical UAV landing speeds to reduce landing errors in turbulent environments. The results have demonstrated that UAVs descending at higher speeds exhibit more stable and concentrated landing trajectories under identical maximum wind speed conditions, and can improve landing success rates by up to 30%. This offers a promising direction for enabling autonomous offshore wind turbine blade inspection, maintenance, and repair by multi-robot platforms.

Higher descent speeds cause a quadratic increase in kinetic energy. In this study, an 18.5 kg UAV descending at 0.5–3.0 m/s generates kinetic energy ranging from 2.3 J to 83.3 J, which may further increase under turbulence. If this energy exceeds the safe threshold for either the UAV or the wind turbine blade, structural damage may occur.

However, the structural interaction between UAVs and turbine blades was not comprehensively considered in this work, nor was the ground effect. Therefore, future investigations should incorporate detailed UAV–blade interaction models and aerodynamic effects near surfaces to define the safe threshold.

Building on this safe threshold, landing gear can be designed to dissipate excess energy, ensuring safer and more accurate landings. Current energy-absorbing systems for UAVs often introduce significant additional mass, which further elevates kinetic energy. As such, the development of lightweight landing gear with high energy absorption capacity presents a promising direction for future research.

Finally, this work was conducted solely in the simulation domain using MATLAB and SIMULINK toolboxes. Experimental validation under real-world field conditions is therefore necessary to confirm the applicability of the findings. Furthermore, the control analysis was limited to a classic PID controller, which constrains the scope of performance evaluation. Future work should explore more advanced controllers, such as Model Predictive Control (MPC), adaptive control, or learning-based approaches, both to rule out potential limitations of relying solely on a PID controller and to assess whether high-speed descent strategies can be extended to more severe turbulent environments.

References

[1] Ren21. *Renewables 2022 Global Status Report*. REN21 Secretariat, Paris, 2022.

[2] HF Zhou, HY Dou, LZ Qin, Y Chen, YQ Ni, and JM Ko. A review of full-scale structural testing of wind turbine blades. *Renewable and Sustainable Energy Reviews*, 33:177–187, 2014.

[3] Zhengyi Jiang, Ferdian Jovan, Peiman Moradi, Tom Richardson, Sara Bernardini, Simon Watson, Andrew Weightman, and Duncan Hine. A multirobot system for autonomous deployment and recovery of a blade crawler for operations and maintenance of offshore wind turbine blades. *Journal of Field Robotics*, 40(1):73–93, 2023.

[4] Yiming Liu, M Hajj, and Y Bao. Review of robot-based damage assessment for offshore wind turbines. *Renewable and Sustainable Energy Reviews*, 158:112187, 2022.

[5] John Bass, Isaac Tunney, and Alexis Lussier Desbiens. Adaptative friction shock absorbers and reverse thrust for fast multirotor landing on inclined surfaces. *IEEE Robotics and Automation Letters*, 7(3):6701–6708, 2022.

http://www.imavs.org/

[6] Bo Hang Wang, Dao Bo Wang, Zain Anwar Ali, Bai Ting Ting, and Hao Wang. An overview of various kinds of wind effects on unmanned aerial vehicle. *Measurement and Control*, 52(7-8):731–739, 2019.

[7] S Caires, J-J Schouten, L Lønseth, V Neshaug, I Pathirana, and O Storas. Uncertainties in offshore wind turbulence intensity. In *Journal of Physics: Conference Series*, volume 1356, page 012037. IOP Publishing, 2019.

[8] Mozhou Gao, Chris H Hugenholtz, Thomas A Fox, Maja Kucharczyk, Thomas E Barchyn, and Paul R Nesbit. Weather constraints on global drone flyability. *Scientific reports*, 11(1):12092, 2021.

[9] Daniel Hentzen, Thomas Stastny, Roland Siegwart, and Roland Brockers. Disturbance estimation and rejection for high-precision multirotor position control. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2797–2804. IEEE, 2019.

[10] Vishnu Saj, Bochan Lee, Dileep Kalathil, and Moble Benedict. Robust reinforcement learning algorithm for vision-based ship landing of uavs. *arXiv preprint arXiv:2209.08381*, 2022.

[11] Xabier Olaz, Daniel Alaez, Manuel Prieto, Jesús Villadangos, and José Javier Astrain. Quadcopter neural controller for take-off and landing in windy environments. *Expert Systems with Applications*, 225:120146, 2023.

[12] Craig WA Murray, Murray Ireland, and David Anderson. On the response of an autonomous quadrotor operating in a turbulent urban environment. In *AUVSI's unmanned systems conference*, 2014.

[13] Kenan Cole and Adam Wickenheiser. Spatio-temporal wind modeling for uav simulations. *arXiv preprint arXiv:1905.09954*, 2019.

[14] Kostas Alexis, George Nikolakopoulos, and Anthony Tzes. Constrained-control of a quadrotor helicopter for trajectory tracking under wind-gust disturbances. In *MELECON 2010-2010 15th IEEE Mediterranean Electrotechnical Conference*, pages 1411–1416. IEEE, 2010.

[15] Davide Todeschini, Lorenzo Fagiano, Claudio Micheli, and Aldo Cattano. Control of vertical take off, dynamic flight and landing of hybrid drones for airborne wind energy systems. In *2019 American control conference (ACC)*, pages 2177–2182. IEEE, 2019.

[16] Vaibhav Ghadiok, Jeremy Goldin, and Wei Ren. On the design and development of attitude stabilization, vision-based navigation, and aerial gripping for a low-cost quadrotor. *Autonomous Robots*, 33(1):41–68, 2012.

[17] Arthur P Mendez, James F Whidborne, and Lejun Chen. Wind preview-based model predictive control of multirotor uavs using lidar. *Sensors*, 23(7):3711, 2023.

[18] Nitin Sydney, Brendan Smyth, and Derek A Paley. Dynamic control of autonomous quadrotor flight in an estimated wind field. In *52nd IEEE Conference on Decision and Control*, pages 3609–3616. IEEE, 2013.

[19] C Galleguillos, A Zorrilla, Antonio Jimenez, L Diaz, ÁL Montiano, M Barroso, Antidio Viguria, and Fernando Lasagni. Thermographic non-destructive inspection of wind turbine blades using unmanned aerial systems. *Plastics, Rubber and Composites*, 44(3):98–103, 2015.

[20] Quan Quan. *Introduction to multicopter design and control*, volume 10. Springer, 2017.

[21] Amrit Shankar Verma, Jiquan Yan, Weifei Hu, Zhiyu Jiang, Wei Shi, and Julie JE Teuwen. A review of impact loads on composite wind turbine blades: Impact threats and classification. *Renewable and Sustainable Energy Reviews*, 178:113261, 2023.

[22] Christian Bak, Frederik Zahle, Robert Bitsche, Taeseong Kim, Anders Yde, Lars Christian Henriksen, Morten Hartvig Hansen, José Pedro Albergaria Amaral Blasques, Mac Gaunaa, and Anand Natarajan. The dtu 10-mw reference wind turbine, 2013. Danish Wind Power Research 2013 ; Conference date: 27-05-2013 Through 28-05-2013.

[23] Ivan Lopez-Sanchez and Javier Moreno-Valenzuela. Pid control of quadrotor uavs: A survey. *Annual Reviews in Control*, 56:100900, 2023.

[24] Frederic M Hoblit. Gust loads on aircraft: concepts and applications. 1988.

[25] Hyeongseok Kim, Daejin Lim, and Kwanjung Yee. Investigation of operable conditions and excess battery energy for quadrotors under wind turbulence. *Journal of Intelligent & Robotic Systems*, 104(3):52, 2022.

[26] US Military Specification: Flying Qualities of Piloted Airplanes. Technical Report MIL-F-8785C, Department of Defense, USA, November 1980. Military Standard.

[27] Siegfried Heier. *Grid integration of wind energy: onshore and offshore conversion systems*. John Wiley & Sons, 2014.

Reinforcement Learning based Optimal Guidance for Landing the Variable Skew Quad Plane on a Ship

Cansu Yıkılmaz*, and Christophe De Wagter†
Delft University of Technology, Delft, The Netherlands

http://www.imavs.org/

Abstract

In this work, we present a reinforcement-learning-based approach for optimal guidance in landing a Variable Skew Quad Plane (VSQP) on a moving ship platform. We develop a reinforcement learning framework that computes the optimal acceleration inputs for the inner adaptive incremental nonlinear dynamic inversion-based controller. Through simulations, we evaluate the performance of various reward function combinations based on the key metrics: touchdown velocity, deviation, and duration. With the optimal reward for the given landing problem, we validate the approach on real ship data. Our results indicate that the reinforcement-learning-based approach outperforms the benchmark linear controller in achieving smoother and safer landings even under complex motion characteristics.

1 Introduction

Unmanned Aerial Vehicles have found use in a wide variety of applications, including environmental monitoring, search and rescue missions, and surveillance [1, 2] thanks to their capability to cost-effectively collect real-time data while being able to operate in remote and hazardous locations. And while sea environments constitute a large part of the planet, operations are often limited since the recovery of the UAV on the ship requires advanced autonomous landing technology.

The complexity of autonomous landing is a challenging, high-performance task. A big part of the challenge comes from the unpredictable motion under the effect of external disturbances and uncertainties [3]. In the case of ship landings, the unpredictability of the sea environment adds an extra layer of difficulty.

An autonomous landing problem has been extensively studied [5, 6, 7, 8, 9, 3] in terms of its sub-phases: target detection, relative state estimation, tracking and landing. The majority of studies rely on vision-based systems for target detection and relative state estimation tasks, with methods differing according to the characteristics of the landing platform (static vs dynamic). Vision-based systems are appreciated in maritime operations [10, 11] since reliance on inertial navigation alone is not feasible.

Concerning control, autonomous landing has relied on a variety of methods, ranging from PID controllers [8, 12, 13], adaptive control techniques [14, 15], robust flight [16] to model predictive control [17, 18] methods. While these techniques are effective in certain cases, they come with their own set of challenges. Classical feedback controllers often face problems with robustness due to dependency on fixed gains and perform poorly under the effect of external disturbances [19]. More advanced controllers like model predictive and robust flight control techniques require a detailed model and obtaining a perfect representation of the real-world system is often challenging. Moreover, the full formulation of the landing as an optimal control problem is numerically heavy. This highlights the need for more efficient and robust solutions in autonomous landing.

Reinforcement learning as a machine learning algorithm has gained significant popularity in recent years and found use in solving guidance and control problems [20, 21]. While optimal control techniques often require complete knowledge of the system dynamics, reinforcement learning is able to learn from raw environmental data without having prior information on the system and adapt to unknown dynamics and unpredictable changes in the environment.

In this work, we use reinforcement learning, particularly Proximal Policy Optimization (PPO), to develop an optimal guidance policy for landing the Variable Skew Quad Plane (VSQP) on a moving platform (ship). We feed the optimal control inputs (accelerations) and feed them back into an adaptive incremental nonlinear dynamic inversion-based controller, which includes a Weighted Least Squares (WLS) based optimization routine for control allocation. We account for the motion in the z-direction and search for the best objective function to minimize landing impacts and lateral offset within a given time. Finally, we test our approach with real-world ship data and compare it with a PID controller.

2 Related Work

Reinforcement learning has become an effective approach for solving autonomous landing problems. One of the earliest studies by Polvara et al. [22] utilized Deep Q-Networks (DQN) for landing on a static platform using low-resolution images coming from a downward-facing camera. Later, Lee et al. [23] developed an actor-critic framework where they included a PID-based inner attitude controller, a ground-

*Email address(es): cnsyklmzoutlook.com
†Email address(es): c.dewagtertudelft.nl



(a) VSQP in quad-mode configuration. (b) VSQP in transition (c) VSQP in forward flight configuration.

Figure 1: The configuration modes of the VSQP: Variable Skew Quad Plane [4]

looking camera model, and a laser rangefinder to enable precise autonomous landing on a static target. The challenge of landing on a dynamic target was addressed by Rodriguez et al. in [24], where they used Deep Deterministic Policy Gradient (DDPG) with Gazeebo. Xie et al. [25] later divided the problem into two parts: perception and relative pose estimation on the one hand, and trajectory optimization and control on the other. In contrast to the previous studies, they also accounted for sensor noise, intermittent measurements, randomness in UAV movement, and incomplete or inaccurate observations. Moreover, they explored a hybrid approach that combined reinforcement learning with heuristic methods for tracking and landing tasks.

In the context of ship landing, Saj et al. [26] claim that previous studies lack the robustness component, which leads to the development of not fully efficient algorithms for ship landing problems. They adapted the twin delayed DDPG (TD3) algorithm for a vision-based ship landing. To overcome the reality gap problem, they applied a domain randomization approach [27] through a variation of environment parameters. They showcased the superior performance of their framework over a PID controller in the real-life tests made under varying wind conditions. These studies collectively highlight the capabilities of reinforcement learning in solving autonomous landing problems.

3 VARIABLE SKEW QUAD PLANE (VSQP)

The Variable Skew Quad Plane is a hybrid UAV designed by the Micro Air Vehicle Laboratory (MAVLab) at TU Delft. The VSQP has two modes of operation - hover and cruise. It is controlled through the skew angle Λ. In hover mode ($\Lambda = 0\,\mathrm{deg}$), the drone operates as a quadrotor with an extra pusher propeller. In the cruise mode ($\lambda = 90$), it operates as a fixed-wing aircraft and achieves forward speed with a push propeller located at the tail. The VSQP, with an intermediate skew angle, resembles an Oblique Flying Wing (OFW) plane. This design offers better gust rejection in hover mode as the wing hides against the fuselage and lower drag during cruise mode since the hover motors hide in the fuselage. The configuration modes of the VSQP are shown in Figure 1.

The guidance and control model of the VSQP incorporates an Adaptive Incremental Nonlinear Dynamic Inversion (ANDI) controller with the Weighted-Least-Square-based control allocation, as the VSQP is an overactuated system [28]. This controller is implemented as a "one-loop" architecture as opposed to cascaded controller approaches in previous ANDI work [29]. This allows the control to distribute the load optimally throughout the control surfaces of the VSQP. The diagram showing the overall guidance and control scheme of the VSQP is given in Figure 2. Within the $G\&C$ structure, the WLS block takes the virtual control parameters calculated by the error controllers for the position and attitude. The reference model is used to shape the given desired setpoint into reference signals that are feasible for the platform.

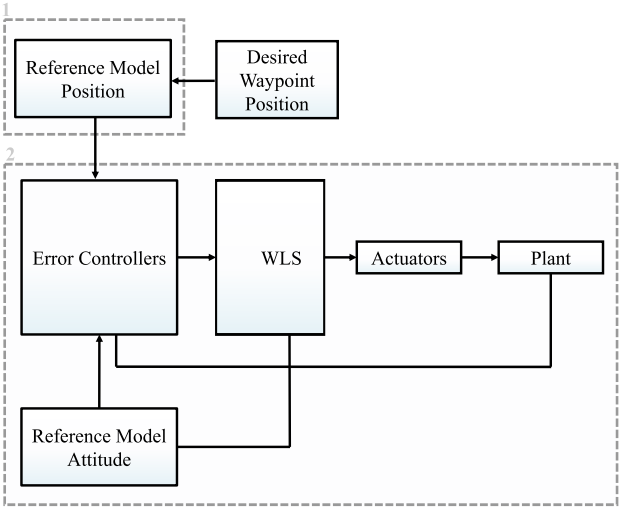


Figure 2: Adaptive Incremental Nonlinear Dynamic Inversion (ANDI) controller with WLS-based control allocation

To perform the optimal path planning, the guidance scheme of the VSQP is replaced by a reinforcement learning-based model to learn the optimal acceleration setpoints. The simplified diagram for the reinforcement learning framework is shown in Figure 4. The reinforcement learning framework contains a deep neural network structure and takes the states of the VSQP as inputs and generates the corresponding optimal acceleration inputs, which are denoted as a_{des}. The response of the vehicle with its ANDI controller, symbolized as

http://www.imavs.org/

http://www.imavs.org/

a_{real}, corresponds to the actual accelerations resulting from the applied control inputs. To effectively map the given commands to the actual accelerations and use them in the training process, a simplified model for both the UAV and the controller is necessary. While previous ANDI work [30] suggests a first-order transfer function for normal quadrotor dynamics, the optimization routine in the VSQP is more complex.

In addition to the existing actuators of the VSQP, the WLS also uses pitch and roll angles as its virtual control inputs within its optimization routine, which are later fed back to the attitude reference model. These Euler angles have the slowest response among all actuators, therefore limiting the vehicle's response to the given control input. A simplified model for the VSQP is therefore obtained by modeling the attitude response of the complete ANDI+VSQP. The result is a transfer function that maps the desired accelerations to the actual ones. It was mathematically derived from the Simulink model and is given in Eq. 1.

$$acc_{tf} = \frac{206s^3+861.1s^2+287s+95.68}{s^6+20.22s^5+143.3s^4+682s^3+1021s^2+340.2s+95.68} \quad (1)$$

The response of the transfer function to a step input is shown in Figure 3. To improve the numerical stability, an equivalent fourth-order model is used to avoid the numerical issues.

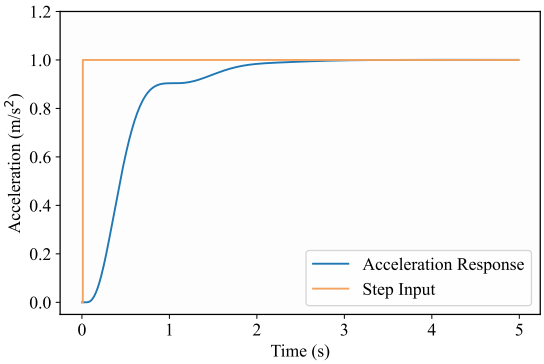


Figure 3: Response of the VSQP to a step input

4 Reinforcement Learning Based Problem Formulation

Reinforcement learning has an agent interact with an environment to take actions that maximize a cumulative reward over time. At each time step t, the agent receives a state s_t from a state space $\mathcal{S}$ and chooses an action a_t from an action space $\mathcal{A}$, based on a policy $\pi(a_t \mid s_t)$. The policy could be stochastic $\pi(a \mid s)$, with a probability associated with each possible action, or deterministic $\pi(s)$, indicating the agent's decision-making process from states to actions. The agent receives a reward r_t for its actions and transitions to the next state s_{t+1}, according to the reward function $\mathcal{R}(s,a)$ and state transition probability $\mathcal{P}(s_{t+1} \mid s_t, a_t)$. This process continues until the agent reaches a terminal state and restarts. The cumulative reward is defined as the return function $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ discounted with a factor $\gamma \in (0,1]$.

The expectation of cumulative reward for a specific policy is represented as V^π, the value function, and is given as:

$$V^\pi(s_t) = \mathbb{E}\left[R_t \mid s_t, a_t = \pi(s_t)\right] \quad (2)$$

The action-value function Q^π in Eq. 3 is equivalent to the value function for every action-state pair (s_t, a_t).

$$Q^\pi(s_t,a_t) = r(s_t,a_t) + \gamma \sum p(s_{t+1} \mid s_t,a_t) V^\pi(s_{t+1}) \quad (3)$$

The goal is to find the optimal policy π^* as in Eq. 4 that maximizes the value function. The corresponding value function is called the optimal value function.

$$\pi^* = \arg\max_\pi V^\pi(s_t) \quad (4)$$

Policy gradient methods are a class of reinforcement learning algorithms that are particularly effective in environments with continuous state and action spaces. In a policy gradient algorithm, the policy is presented as π_θ where θ are neural network parameters. Given that the expected return is $J(\theta) = \mathbb{E}_s[V_{\pi_\theta}(s)]$, the goal is to find optimal parameters $\theta^* = \arg\max_\theta J(\theta)$ through a gradient ascent update $\theta_{k+1} = \theta_k + \alpha_k \nabla J(\theta_k)$ where α_k is the learning rate. The gradient of $J(\theta)$ is calculated using the policy gradient theorem, which is given by:

$$\nabla J(\theta) = \mathbb{E}_{\pi_\theta}\left[Q_{\pi_\theta}(s,a) \nabla \log \pi_\theta(s,a)\right] \quad (5)$$

There are many types of policy gradient algorithms, such as TRPO [31], SAC [32], and PPO [33]. In this work, we adapt the PPO algorithm to solve the landing problem of the VSQP on a moving platform. PPO is a highly stable and efficient policy gradient algorithm that introduces techniques like clipping the objective and using an adaptive penalty coefficient to improve the stability and efficiency of the learning process [33]. It employs an actor-critic model where the actor takes output actions and the critic evaluates its performance.

PPO uses the following objective:

$$L^{CLIP}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right] \quad (6)$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ represents the probability ratio of the action under the new and old policies, $\hat{A}_t$ is advantage at timestep t, and ϵ is a hyperparameter that defines the clipping range to keep the ratio within reasonable bounds.

Problem Formulation

The UAV landing problem can be described as a Markov decision process (MDP). Typical MDP can be defined as a four tuple $(\mathcal{S}, \mathcal{A}, R, \Omega)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $R(s,a)$ represents the immediate rewards, and

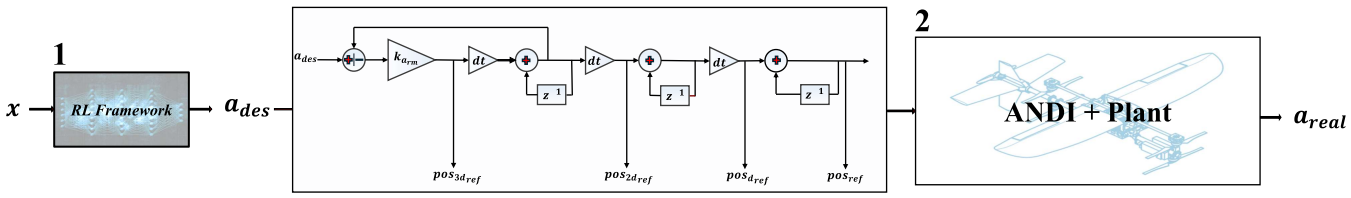


Figure 4: Reinforcement learning framework integrated into the existing control structure

Ω is the observation space. Based on this model, the landing problem is formulated as:

1. *State Space* $\mathbb{S}_d$: The state space $\mathbb{S}_d$ represents the states of the UAV. Give the 4th order transfer function model of ANDI+VSQP, the state space for the drone consists of extra three acceleration derivatives $\dot{\boldsymbol{a}}_d, \ddot{\boldsymbol{a}}_d, \dddot{\boldsymbol{a}}_d$ on top of the position $\boldsymbol{p}_d$, velocity $\boldsymbol{v}_d$, and acceleration $\boldsymbol{a}_d$ components:

$$\mathbb{S} = \{\boldsymbol{p}_d, \boldsymbol{v}_d, \boldsymbol{a}_d, \dot{\boldsymbol{a}}_d, \ddot{\boldsymbol{a}}_d, \dddot{\boldsymbol{a}}_d\} \tag{7}$$

2. *State Space* $\mathbb{S}_s$: The state space $\mathbb{S}_s$ represents the states of the ship platform. It consists of platform position p_s and velocity v_s components:

$$\mathbb{S}_s = \{\boldsymbol{p}_s, \boldsymbol{v}_s\} \tag{8}$$

The motion of the ship is characterized by sinusoidal signals that are governed by specific frequencies, amplitudes, and initial conditions. The definitions for position and velocity are defined as:

$$p_i(t) = A_{\mathrm{p},i} \sin(\omega_i t + \phi_i) + p_{\mathrm{init},i} + \xi_i(t) \tag{9}$$

$$v_i(t) = \omega_i A_{\mathrm{p},i} \cos(\omega_i t + \phi_i) + \eta_i(t) \tag{10}$$

where $A_{\mathrm{p},i}$ is the amplitude of the sinusoidal motion for each axis i, ω_i denotes the angular frequency, ϕ_i is the phase shift, and finally $\eta_i(t)$ captures the random walk component. This random component helps to simulate unpredictable movements or external disturbances affecting the ship's motion.

3. *Observation Space* Ω: The input to the reinforcement learning algorithm is given as the relative position p_r and velocity v_r components of the drone with respect to the moving platform, along with the *real acceleration* $\boldsymbol{a}_d$ of the drone:

$$\mathbb{S}_s = \{\boldsymbol{p}_r, \boldsymbol{v}_r, \boldsymbol{a}_d\} \tag{11}$$

4. *Action Space* $\mathbb{A}$: The action space Eq. 12 $\mathbb{A}$ consists of *desired acceleration* components $a_{n_{des}}, a_{e_{des}}, a_{d_{des}}$ determined based on the inputs of the reference model:

$$\mathbb{A} = \{a_{n_{des}}, a_{e_{des}}, a_{d_{des}}\} \tag{12}$$

Within the reinforcement learning framework, the control input parameters a_n and a_e are used to steer the drone onto the platform, while a_d is mainly used to find the right time to accomplish landing.

5. *Reward Function* R: Designing a reward function that is suitable for a *landing on a moving target* problem is not a straightforward task. Several options are defined and we classify the reward function in terms of three components: position tracking, velocity tracking, and collision penalty.

   (a) Position Tracking: The reward function R_p in Eq. 13 is designed for tracking of the moving platform by the minimization of the deviation in both vertical and horizontal planes relative to the target:

$$R_p = k_1 \left|p_{r_{\mathrm{old}_v}}\right| - k_1 \left|p_{r_{\mathrm{new}_v}}\right| + k_2 \left\|\mathbf{p}_{r_{\mathrm{old}_h}}\right\| - k_2 \left\|\mathbf{p}_{r_{\mathrm{new}_h}}\right\| \tag{13}$$

   The parameters $p_{r_{\mathrm{old}_v}}$ and $p_{r_{\mathrm{new}_v}}$ represent the old and new vertical distances; $\mathbf{p}_{r_{\mathrm{old}_h}}$ and $\mathbf{p}_{r_{\mathrm{new}_h}}$ represent the horizontal distances between the UAV and the target, respectively. The term $\left|p_{r_{\mathrm{old}_v}}\right| - \left|p_{r_{\mathrm{new}_v}}\right|$ rewards the agent based on the absolute difference in vertical position and $\|\mathbf{p}_{r_{\mathrm{new}_v}}\| - \|\mathbf{p}_{r_{\mathrm{new}_h}}\|$ uses the norm of both x and y components together to compute the difference. The relative importance is defined by the weights k_1 and k_2, which are determined to be 2 and 1, respectively.

   (b) Velocity Tracking: The velocity reward R_v is designed to ensure that the speed of the UAV is adjusted in a way that it slows down while it gets closer to the ship. Similarly to the position reward R_p, the parameters $v_{r_{\mathrm{old}_v}}$ and $v_{r_{\mathrm{old}_h}}$ represent

http://www.imavs.org/

the old and new relative velocities in the vertical plane, whereas $\mathbf{v}_{r_{\text{old}_\text{v}}}$ and $\mathbf{v}_{r_{\text{old}_\text{h}}}$ represent the corresponding parameters in the horizontal plane. The reward function is dynamically scaled by a factor that depends on the drone's height above the ship.

$$R_v = k_1 \left(v_{r_{\text{old}_\text{v}}} - v_{r_{\text{new}_\text{v}}}\right) \left(\frac{h_t + h}{h_t}\right) + k_2 \left(v_{r_{\text{old}_\text{h}}} - v_{r_{\text{new}_\text{h}}}\right) \left(\frac{h_t + h}{h_t}\right) \quad (14)$$

(c) Collision Penalty: The collision penalty is designed to directly penalize the UAV's speed at the touchdown point. To assess the performance of the landing, two different reward functions are considered based on relative velocities and absolute velocities.

The first reward function R_{c_1} as given in Eq.15 is defined as the *relative velocity collision penalty* that penalizes the relative velocity of the UAV with respect to the platform. The constants k_3 and k_4 allow tuning the sensitivity of the penalty to the different directional velocities, avoiding lateral collisions as well as maintaining a safe descent speed.

$$R_{c_1} = k_3 \left|v_{r_{\text{new}_v}}\right|^2 + k_4 \left\|\mathbf{v}_{r_{\text{new}_h}}\right\|^2 \quad (15)$$

The second collision reward R_{c_2}, *absolute velocity collision penalty*, focuses on the vertical velocity components at the point of landing. The $v_{d_{\text{new}_z}}$ indicates the vertical velocity of the drone, and $v_{p_{\text{new}_z}}$ refers to the platform's vertical velocity.

$$R_{c_2} = k_5 \left|v_{d_{\text{new}_z}}\right|^2 + k_6 \left|v_{p_{\text{new}_z}}\right|^2 \quad (16)$$

These collision rewards only get active at the final point of landing. This approach resembles a form of Cliffwalk [34] since the agent does not receive intermediate rewards till the point of touchdown. While this makes the search for an optimal policy more challenging, the agent gains greater freedom in its actions as opposed to the reward functions used in the previous studies [24, 26, 25], which defined a separate reward function for each altitude range.

These reward functions are strategically combined to identify the most effective objective for the given problem. The table 1 outlines the combinations used in this study. Reward R1 only consists of the position reward R_p, encouraging the agent to minimize the difference between the old and new positions. R2 and R3 incorporate the collision penalties R_{c_1} and R_{c_2} along with the position reward. Lastly, rewards R4 and R5 combine everything, including the velocity reward R_v, with the collision penalties.

Table 1: Reward combinations used in the study

Reward Number	Combination
R1	R_p
R2	$\text{R}_p + \text{R}_{c_1}$
R3	$\text{R}_p + \text{R}_{c_2}$
R4	$\text{R}_p + \text{R}_v + \text{R}_{c_1}$
R5	$\text{R}_p + \text{R}_v + \text{R}_{c_2}$

5 TRAINING

For training the reinforcement learning framework, a gym environment including classes for both the UAV and the ship platform dynamics was created in Python.

The ship model uses sinusoidal signals with mixed frequencies and random walk components to simulate a wide variety of signals that resemble real ship motion. This variety in the ship motion is critical to prevent overfitting during the training process. In alignment with the observation space Ω from Eq.11, the relative position, relative velocity, and real drone accelerations are given to the model at every time step. The outputs are the desired accelerations, which are later used to propagate the states of the drone.

Since the ship is moving at a slow constant speed and this study focuses on relative landing control, the constant forward motion of the ship is ignored, and only the wave-induced hard-to-predict random motions are modeled. In the horizontal plane, the relative navigation of the UAV is evaluated by selecting different UAV initial conditions. The training process is repeated for all the combinations indicated in Table 1. The simulations are ended when any of the following conditions are met: a ground collision occurs, the drone exits the designated area, the maximum number of steps is reached, or the drone achieves a landing condition of being below 0.1m above the platform.

5.1 Initializing the drone dynamics

For training, a vectorized environment is used to run several environments in parallel. To do so, initial drone states are uniformly sampled from the following integrals:

$$\begin{array}{lll} p_n \in [-1,1] + p_{n_i} & p_e \in [-1,1] + p_{e_i} & p_d \in [-1,1] + p_{d_i} \\ v_n \in [-0.5, 0.5] & v_e \in [-0.5, 0.5] & v_d \in [-0.5, 0.5] \\ a_n \in [-0.05, 0.05] & a_e \in [-0.05, 0.05] & a_d \in [-0.05, 0.05] \end{array} \quad (17)$$

Here, $p_{n_i}, p_{e_i}, p_{d_i}$ represent the initial position elements of the drone. The parameters p_{n_i} and p_{e_i} are chosen in a way that the drone aligns itself with the platform and p_{d_i} is determined to be 10 meters as the starting altitude at the

http://www.imavs.org/

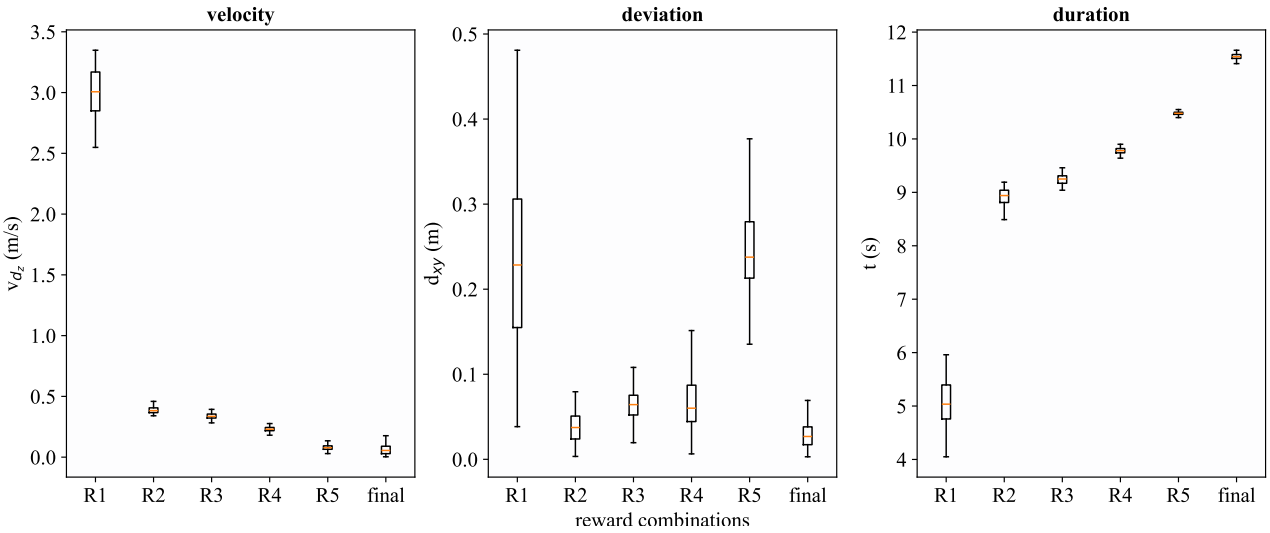


Figure 5: The analysis of reward combinations in terms of performance metrics

http://www.imavs.org/

beginning of the training procedure. Additionally, nine additional state elements which are the derivatives of accelerations ($\dot{\boldsymbol{a}}_d, \ddot{\boldsymbol{a}}_d, \dddot{\boldsymbol{a}}_d$) up to a third degree are also initialized with a value of zero with the given state elements.

5.2 *Initializing the ship dynamics*

Similarly to the drone model, ship states are uniformly sampled from the intervals:

$$\begin{array}{lll} A_x \in [-0.1, 0.1] & A_y \in [-0.1, 0.1] & A_z \in [-0.1, 0.1] + A_z \\ f_x \in [-0.1, 0.1] & f_y \in [-0.1, 0.1] & f_z \in [-0.1, 0.1] + f_z \end{array} \quad (18)$$

For the ship, wave parameters amplitude A and frequency f are used to initialize the sinusoidal motion characteristics. To create variation in the generated sinusoidal motions additional parameters such as mix frequency f_{mix}, random walk step size rw_{ts}, and low pass filter coefficient α are also given as an input to the model. The values for these parameters are determined as 0.2, 0.01, and 0, respectively.

6 Evaluation

The performance of reward functions from Table 1 is assessed based on three key metrics: touchdown velocity, deviation, and duration. This evaluation is completed for a specific test case scenario defined as follows. A sinusoidal wave in the z-direction is generated using the characteristics:

$$A_z \in [-0.1, 0.1] + A_z, \quad f_z \in [-0.1, 0.1] + f_z \quad (19)$$

The touchdown velocity is a critical metric for assessing the safety and smoothness of the UAV's landing. Too high velocities result in UAV damage or even a crash. The deviation metric measures the accuracy of the UAV's landing by evaluating how close the UAV lands to the intended target point. Lastly, time to collision represents the duration that a landing takes.

The assessment of the drone is visualized with box plots in Figure 5. The left subplot shows the distribution of touchdown velocities for each reward combination. The y-axis represents the vertical descent velocity v_{d_z} in meters per second (m/s), and the x-axis lists the reward combinations (R1 to R5) and the final (R5 with tuned coefficients). R1 has the highest median touchdown velocity at around 3.0 m/s. In contrast, the value of the median velocity drops significantly for R2 and R3, while the latter has a slightly lower median value. This shows that having a collision penalty, either absolute or relative, has a significant effect in reducing touchdown velocity. With the addition of the velocity element in R4 and R5, the velocity values further decrease, with touchdown velocities close to zero, indicating the landing is extremely smooth and controlled.

The middle box plot shows the deviation from the target center (d_{xy}) in meters for each reward combination. While the deviation values are all within landing margins, for rewards R2, R3, and R4, the deviation is less, while R1 and R5 show that the drone deviates a bit more from the center. Finally, the right plot shows the landing duration in seconds. As expected, the duration gets longer as the velocity values decrease, resulting in a less aggressive flight.

While different performance parameters can be of importance, in practical flights, more importance is put on having lower velocities. Based on the results given in the top-left figure, the reward function R5, which uses the absolute velocities of the drone, has been identified as having the optimal form for the given problem.

Further analysis was conducted to determine the effect of coefficients (k_5 and k_6) on the UAV's performance, which was previously given a value of 1. Both coefficients k_5 and

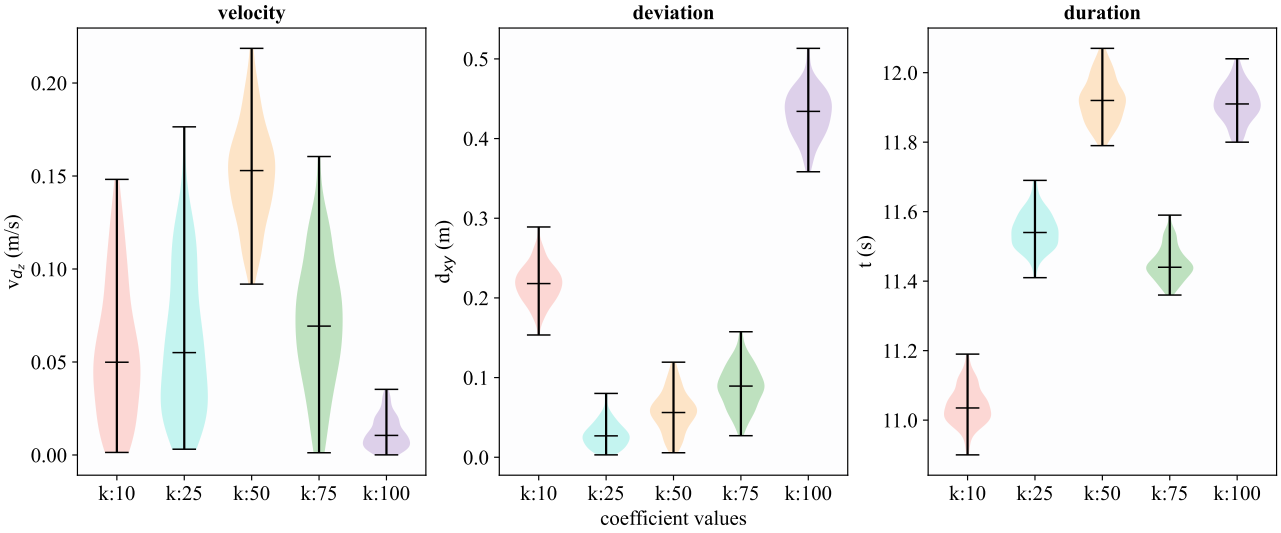


Figure 6: The analysis of reward combinations in terms of performance metrics

http://www.imavs.org/

k_6 are tested with the values of 10, 25, 50, 75, and 100, respectively. The results are given in the Figure 6, which show the effect of the different coefficients on the UAV's performance, for the chosen reward combination.

The first plot on the left again shows the distribution of touchdown velocities, but this time for the same combination, R5. Among the coefficients, k:100 achieves the lowest median value of a mere 0.01 m/s. It was also observed that much higher values than 100 either lead to extremely long flights or no landing at all. Specifically, high penalty values for the collision make the drone extremely hesitant to take any action. These are left out of the plot as they were deemed unsuccessful.

The graphs indicate that coefficients between 10 and 100 provide a good balance in terms of performance metrics, but do not show a very linear trend. Considering the negative effect of higher penalties on performance and the low deviation values, the coefficient k:25 was chosen for R5. The final form of the reward function is thus given as:

$$\begin{aligned} R_{final} = & \left|p_{r_{\text{old}_v}}\right| - \left|p_{r_{\text{new}_v}}\right| + 2\left\|\mathbf{p}_{r_{\text{old}_h}}\right\| - 2\left\|\mathbf{p}_{r_{\text{new}_h}}\right\| \\ & + \left(v_{r_{\text{old}_v}} - v_{r_{\text{new}_v}}\right)\left(\frac{h_t + h}{h_t}\right) \\ & + 2\left(v_{r_{\text{old}_h}} - v_{r_{\text{new}_h}}\right)\left(\frac{h_t + h}{h_t}\right) \\ & + 25\left|v_{d_{\text{new}_z}}\right|^2 + 25\left|v_{p_{\text{new}_z}}\right|^2 \end{aligned} \tag{20}$$

7 Validation

To determine if the reinforcement framework could successfully land on waves, validation with real ship data is performed. Real ship data was gathered at a frequency of 5 Hz and interpolated using a quadratic fit to the 10 Hz used in previous simulations. A sample portion is shown in Figure 7.

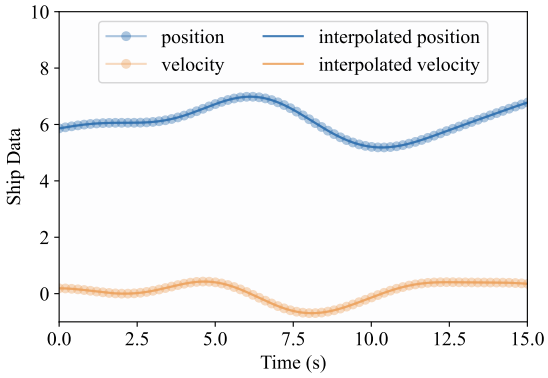


Figure 7: Measured ship motion data for validation

The performance of the reinforcement learning framework was tested on the real ship data using the final form of the reward function (see Eq. 20). Additionally, the RL framework was compared against two benchmark controllers.

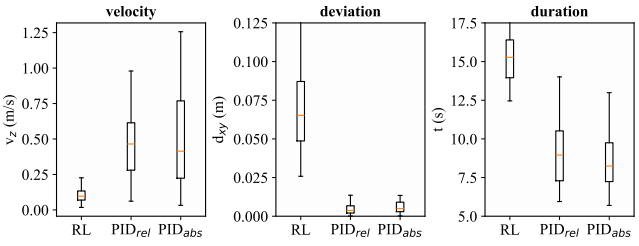


Figure 8: Validation box plots

Figure 8 shows the comparison with PID-based controllers that were designed for position and velocity track-

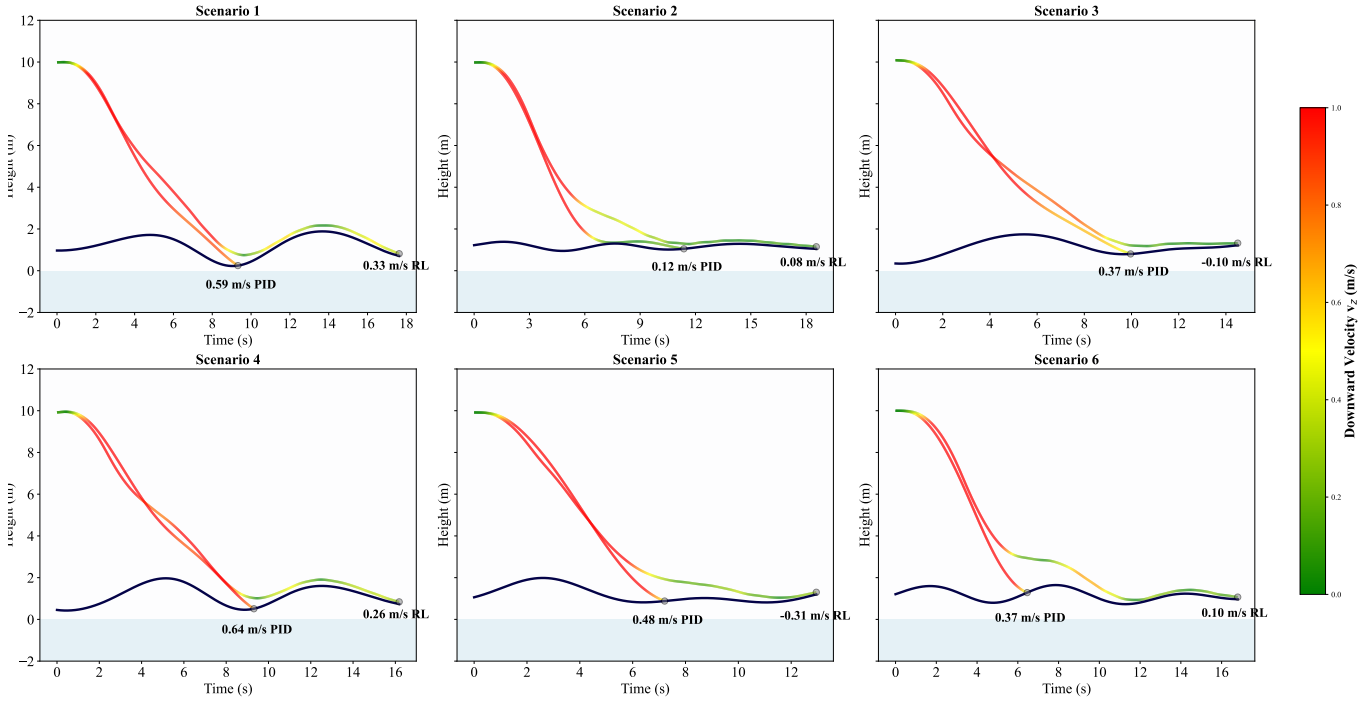


Figure 9: Validation trajectories

ing, namely, PID$_{rel}$ and PID$_{abs}$. The former takes the relative velocity of the drone with respect to the platform, while the latter generates velocity commands based on the position tracking only.

The left plot in Figure 8 compares the vertical descent velocities (v_{d_z}) of the RL framework and the two PID controllers. While the figures show that PID$_{rel}$ works slightly better than the PID$_{abs}$ in terms of downward velocity, the RL framework achieves the lowest median velocity among them, with values close to 0.1 m/s. As the limit for damage sits at around 0.5 m/s, these results indicate that RL shows a superior performance with much smoother landings, even leaving some margin in velocity. The statistics for the reinforcement learning and PID controllers are given in Table 2.

Finally, Figure 9 visualizes the trajectories of a drone controlled by both PID controllers (PID$_{rel}$) and the reinforcement learning framework using real ship data not seen during training across six different scenarios. The y-axis indicates the height (in meters) and the x-axis shows the duration of the flight (in seconds). The trajectories are color-coded to indicate the downward velocity (v_{d_z}) of the drone, with green representing low velocities and red representing high velocities. The impact velocities are indicated for both controllers.

The trajectories controlled by the PID consistently result in landings with high touchdown velocities, despite having relative velocity control. In each scenario, PID tries to slow down when approaching the ship, but does not always succeed in adapting to the ship's motion due to its feedback control structure and UAV inertial. In contrast, the reinforcement learning framework displays significantly different landings. It closely tracks the wave patterns of the ship and appears to wait for good conditions before landing. It consistently shows lower touchdown velocities with stable and controlled flights as opposed to the PID control that often results in impact velocities over 0.5m/s instead of successful landings.

Table 2: Summary Statistics for RL and PID Controllers

Label	Min	Max	Mean	Std Dev
RL-vz	0.0176	0.2265	0.1017	0.0453
PID$_{rel}$-vz	0.0617	1.3303	0.4779	0.2459
PID$_{abs}$-vz	0.0325	1.2563	0.4828	0.3082
RL-xy	0.0258	0.1320	0.0692	0.0243
PID$_{rel}$-xy	0.0001	0.0433	0.0073	0.0091
PID$_{abs}$-xy	0.0002	0.0453	0.0084	0.0097
RL-ttc	12.4600	19.7600	15.3107	1.6582
PID$_{rel}$-ttc	5.9500	14.0100	9.0416	1.9637
PID$_{abs}$-ttc	5.700	12.990	8.5610	1.7039

8 Conclusion

This study demonstrated the effectiveness of a reinforcement learning-based framework for the autonomous landing of the Variable Skew Quad Plane on a moving ship. Several simulations performed with randomized sinusoidal signals revealed that the reinforcement learning could adapt to a wide variety of sinusoidal ship motions and generate realistic acceleration commands that effectively adjust the drone's speed

http://www.imavs.org/

and trajectory to achieve successful landings.

The validation with real ship data, which incorporated non-sinusoidal and unpredictable movements, confirmed the robustness and adaptability of our framework. Compared to the benchmark controller, reinforcement learning achieved significantly lower impact velocities that resulted in controlled, safer landings, closely tracking the wave patterns of the ship and waiting for an acceptable landing moment.

This work highlighted the capabilities of reinforcement learning for a challenging autonomous landing task, which requires enhanced performance against environmental variability and operational uncertainty. The integration of reinforcement learning as a guidance model within the control loop promotes safer and more reliable autonomous operations.

Future work may expand upon this study by performing real-life experiments and especially investigating the robustness of the proposed algorithm in various sea states. In addition to that, different network structures (e.g., Recurrent Neural Networks) could enhance the results by better learning the patterns in large amounts of sea state data. Lastly, a more detailed analysis of the reward elements may be performed to incorporate ship characteristics further into the objective.

http://www.imavs.org/

REFERENCES

[1] JeongWoon Kim, Yeondeuk Jung, Dasol Lee, and David Hyunchul Shim. Outdoor autonomous landing on a moving platform for quadrotors using an omnidirectional camera. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1243–1252. IEEE, 2014.

[2] Kun Li, Peidong Liu, Tao Pang, Zhaolin Yang, and Ben M Chen. Development of an unmanned aerial vehicle for rooftop landing and surveillance. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 832–838. IEEE, 2015.

[3] Yi Feng, Cong Zhang, Stanley Baek, Samir Rawashdeh, and Alireza Mohammadi. Autonomous landing of a uav on a moving platform using model predictive control. *Drones*, 2(4):34, 2018.

[4] Tomaso Maria Luigi De Ponti. Incremental nonlinear dynamic inversion controller for a variable skew quad plane. Master's thesis, Delft University of Technology, 2022.

[5] Oualid Araar, Nabil Aouf, and Ivan Vitanov. Vision based autonomous landing of multirotor uav on moving platform. *Journal of Intelligent & Robotic Systems*, 85:369–384, 2017.

[6] Liguo Tan, Juncheng Wu, Xiaoyan Yang, and Senmin Song. Research on optimal landing trajectory planning method between an uav and a moving vessel. *Applied Sciences*, 9(18):3708, 2019.

[7] Pablo R Palafox, Mario Garzón, João Valente, Juan Jesús Roldán, and Antonio Barrientos. Robust visual-aided autonomous takeoff, tracking, and landing of a small uav on a moving landing platform for life-long operation. *Applied Sciences*, 9(13):2661, 2019.

[8] Ajmal Hinas, Jonathan M Roberts, and Felipe Gonzalez. Vision-based target finding and inspection of a ground target using a multirotor uav system. *Sensors*, 17(12):2929, 2017.

[9] Marcin Skoczylas. Vision analysis system for autonomous landing of micro drone. *acta mechanica et automatica*, 8(4):199–203, 2014.

[10] Bochan Lee, Vishnu Saj, Dileep Kalathil, and Moble Benedict. Intelligent vision-based autonomous ship landing of vtol uavs. *Journal of the American Helicopter Society*, 68(2):113–126, 2023.

[11] Xuancen Liu, Shifeng Zhang, Jiayi Tian, and Longbin Liu. An onboard vision-based system for autonomous landing of a low-cost quadrotor on a novel landing pad. *Sensors*, 19(21):4703, 2019.

[12] Lizhen Wu, Chang Wang, Pengpeng Zhang, and Changyun Wei. Deep reinforcement learning with corrective feedback for autonomous uav landing on a mobile platform. *Drones*, 6(9):238, 2022.

[13] Lirong Zhou, Anton Pljonkin, and Pradeep Kumar Singh. Modeling and pid control of quadrotor uav based on machine learning. *Journal of Intelligent Systems*, 31(1):1112–1122, 2022.

[14] Botao Hu, Lu Lu, and Sandipan Mishra. Fast, safe and precise landing of a quadrotor on an oscillating platform. In *2015 American Control Conference (ACC)*, pages 3836–3841. IEEE, 2015.

[15] JeongWoon Kim, YeonDeuk Jung, DaSol Lee, and David Hyunchul Shim. Landing control on a mobile platform for multi-copters using an omnidirectional image sensor. *Journal of Intelligent & Robotic Systems*, 84:529–541, 2016.

[16] Shyh-Pyng Shue and Ramesh K. Agarwal. Design of automatic landing systems using mixed h2/h1 control. *Journal of Guidance, Control, and Dynamics*, 22(1), 1999.

[17] Yi Feng, Cong Zhang, Stanley Baek, Samir Rawashdeh, and Alireza Mohammadi. Autonomous landing of a uav on a moving platform using model predictive control. *Drones*, 2(4), 2018.

[18] Alireza Mohammadi, Yi Feng, Cong Zhang, Samir Rawashdeh, and Stanley Baek. Vision-based autonomous landing using an mpc-controlled micro uav on a moving platform. In *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 771–780, 2020.

[19] Man Yuan, Chang Wang, Pengpeng Zhang, and Changyun Wei. *PID with Deep Reinforcement Learning and Heuristic Rules for Autonomous UAV Landing*, pages 1876–1884. Springer Singapore, Singapore, 03 2023.

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[21] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[22] Riccardo Polvara, Massimiliano Patacchiola, Sanjay Sharma, Jian Wan, Andrew Manning, Robert Sutton, and Angelo Cangelosi. Toward end-to-end control for uav autonomous landing via deep reinforcement learning. In *2018 International conference on unmanned aircraft systems (ICUAS)*, pages 115–123. IEEE, 2018.

[23] Seongheon Lee, Taemin Shim, Sungjoong Kim, Junwoo Park, Kyungwoo Hong, and Hyochoong Bang. Vision-based autonomous landing of a multi-copter unmanned aerial vehicle using reinforcement learning. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 108–114. IEEE, 2018.

[24] 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*, 93:351–366, 2019.

[25] Jingyi Xie, Xiaodong Peng, Haijiao Wang, Wenlong Niu, and Xiao Zheng. Uav autonomous tracking and landing based on deep reinforcement learning strategy. *Sensors*, 20:5630, 2020.

[26] Vishnu Saj, Bochan Lee, Dileep Kalathil, and Moble Benedict. Robust reinforcement learning algorithm for vision-based ship landing of uavs. *arXiv preprint arXiv:2209.08381*, 2022.

[27] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.

[28] TML De Ponti, EJJ Smeur, and BWD Remes. Incremental nonlinear dynamic inversion controller for a variable skew quad plane. In *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 241–248. IEEE, 2023.

[29] HJ Karssies and C De Wagter. Extended incremental non-linear control allocation (xinca) for quadplanes. *International Journal of Micro Air Vehicles*, 14:17568293211070825, 2022.

[30] Robin Ferede, Guido de Croon, Christophe De Wagter, and Dario Izzo. End-to-end neural network based optimal quadcopter control. *Robotics and Autonomous Systems*, 172:104588, 2024.

[31] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[32] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

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

[34] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay, 2016.

http://www.imavs.org/

A LiDAR-Based Deep Reinforcement Learning Autonomous Navigation System in Unknown Environments

Yuki Onda, Abner Asignation Jr., and Satoshi Suzuki*
Chiba University, Japan

http://www.imavs.org/

Abstract

This study develops an autonomous, UAV navigation system based on an end-to-end deep reinforcement learning algorithm. While conventional autonomous flight systems rely on constructing environmental maps or prior knowledge, the proposed method directly utilizes obstacle range data from a LiDAR sensor, along with the UAV's own position and velocity information, as inputs to the Proximal Policy Optimization (PPO) algorithm. The policy is trained to derive optimal flight paths based on real-time environmental perception, enabling navigation to target locations without building any environmental maps. To evaluate the effectiveness of the proposed method, flight simulations were conducted in a physics-based simulator featuring obstacle environments. Furthermore, indoor real-world experiments confirmed that the trained policy demonstrated high adaptability and robustness in actual environments. This study contributes to the realization of autonomous flight in more realistic and complex environments without the need for environmental maps, laying the groundwork for future practical applications.

1 Introduction

In recent years, Unmanned Aerial Vehicle (UAV) technology has advanced significantly. UAVs offer several advantages over manned aircraft, including smaller size, lower cost, and ease of operation. Moreover, UAVs are particularly well-suited for complex mission environments such as forest search and rescue operations [1] and bridge inspections [2]. However, such environments often present challenges, including scattered obstacles and unreliable communication or satellite signals [3]. In particular, efficient obstacle avoidance and rapid target acquisition are critical in search and rescue missions. Therefore, navigation capabilities such as autonomous navigation and obstacle avoidance are essential for next-generation UAV systems [4, 5].

Conventional non-learning-based methods suffer from limitations such as the need for explicit path planning and inefficient use of computational resources and memory [6, 7]. To address these issues, a previous study proposed a novel learning-based and model-free approach to UAV navigation in unknown and complex environments [8]. This approach utilizes reinforcement learning, which enables the UAV to learn from patterns and make informed decisions based on data [9]. As a result, the UAV can learn optimal control policies from flight data and reach its destination quickly and safely while avoiding various obstacles. However, in that study, both the UAV and obstacle positions were predefined in simulation. The real-world experiments simply replicated the simulated environment by placing obstacles at the same coordinates as in the simulation. Therefore, the experiments were limited in scope and did not demonstrate the ability to generalize to unknown real-world environments.

In this study, we envision future applications in complex environments such as forests, where visibility is poor and the use of GNSS or vision sensors is difficult. To this end, we extend previous models by incorporating a LiDAR sensor and develop an end-to-end navigation system that enables a drone to reach target locations while avoiding unknown obstacles through reinforcement learning and LiDAR. Furthermore, we verify the effectiveness of the proposed system through both simulations and real-world experiments, while evaluating its generalization performance in environments with unseen obstacles

The organization of this paper is as follows. Section 2 describes an overview of the system developed in this study. Section 3 explains various aspects of the reinforcement learning design used in the simulation. Section 4 presents the simulation environment, experimental setup, and discusses the results and their analysis. Finally, Section 5 concludes the paper and outlines future work.

2 Navigation system using deep reinforcement learning

As shown in 1, this study proposes a guidance framework based on direct policy search and optimization through reinforcement learning. The aim of this framework is to enable obstacle-avoidance navigation for UAVs to reach target destinations in unknown environments. In this framework, low-level angular velocity, attitude, and speed are controlled by a well-tuned cascading PID, while high-level path planning and position control are integrated into a control policy.

*Email address(es): suzuki-s@chiba-u.jp

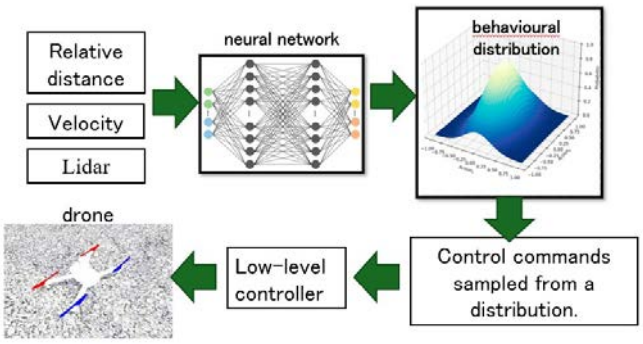


Figure 1: RL controller for learning

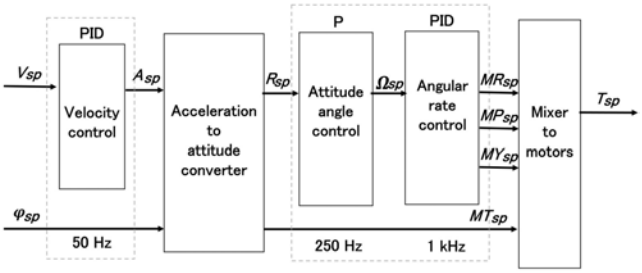


Figure 2: Low level controller

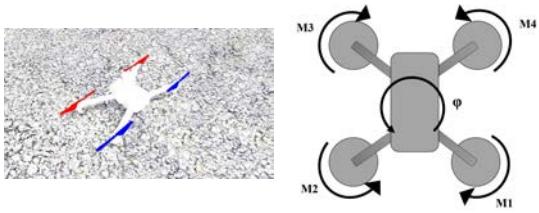


Figure 3: Drone model

This control policy is implemented as a fully connected neural network that takes as input raw sensor measurements from the LiDAR, together with additional state information such as the current UAV velocity, the relative position to the target, and the previous action. The network outputs an action distribution that represents the probability distribution over target speed commands. Specifically, the actor network produces the mean vector of a Gaussian distribution, while the covariance matrix is fixed to a constant value ($0.6I$). Thus, the policy can be expressed as

$$a \sim \pi_\theta(a \mid s) = \mathcal{N}\big(\mu_\theta(s),\, 0.6I\big) \tag{1}$$

This design allows stochastic sampling during training to ensure exploration, although the variance remains constant throughout learning. Finally, the actual action is sampled from this distribution and sent to the low-level controller, enabling both exploration during training and robust execution during deployment.

2.1 Low-Level Controller

In this study, a cascaded control architecture combining P and PID controllers operating at different frequencies is employed as the low-level control system to accurately execute the velocity commands from the high-level RL controller. As shown in Figure 2, the outer loop computes the desired acceleration from the target velocity, which is then converted into the desired attitude angles in the intermediate layer. The inner loop subsequently controls the attitude angles and angular rates. The thrust commands in each direction are finally transformed into motor rotational speeds, enabling stable and responsive flight control. Each control loop operates at a different frequency according to its responsiveness: 50 Hz for velocity control, 250 Hz for attitude control, and 1 kHz for angular rate control.

2.2 High-Level Controller

This study aims to address the problem of high latency in environment recognition and to maximize the agile maneuverability of UAVs through path planning in complex environments. To achieve this, we utilize the Proximal Policy Optimization (PPO) algorithm [10], a type of reinforcement learning method [11], to process environmental and UAV state information collected in parallel from both onboard and external sensors. In the proposed system, the PPO algorithm generates control commands (i.e., velocity commands) based on inputs such as the relative position to the target point and the UAV's velocity, obtained from either the simulator or the motion capture system, as well as obstacle range data from the LiDAR sensor. These commands are then directly supplied to the low-level control system in an end-to-end manner.

3 Simulation

3.1 Problem Setup

The Gazebo simulator, Robot Operating System (ROS), and reinforcement learning algorithm run on a desktop computer running Ubuntu 20.04. The main hardware configuration is an Intel i9-14900K CPU, Nvidia 4080 GPU, and 32GB RAM. In this study, we used the model (Figure 3) from [12], which was designed for simulation in ROS and Gazebo and a 2D LiDAR sensor was used to obtain a point cloud of the forward range except for the backward 90°at a distance of up to 5m from the UAV.

3.2 Hyperparameter

The hyperparameters in Table 1 were used to learn the control policies in all designed learning environments.

3.3 State, Action

The input state to reinforcement learning is $s = [s^{dis}, s^{vel}, s^{lidar}, act_{(t-1)}]$. Here, s^{dis} denotes the relative

http://www.imavs.org/

http://www.imavs.org/

position to the target point, represented by the differences in the x and y coordinates ($\Delta x, \Delta y$), expressed in the global frame (i.e., world coordinates). This requires global localization, which is provided by the OptiTrack system in real-world experiments and by the simulator during training. s^{vel} is the velocity of the UAV in the x and y directions, and s^{lidar} is the range data from the LiDAR sensor, consisting of 417 points. In addition, act, the output of the network one step before, is added as a state quantity to stabilize the output of the network. Here, the network output, action act, is a velocity command to the low-level controller in the x- and y-directions. In order to ensure stable learning, the state variables S fed into the network are normalized within the range of -1 to 1.

3.4 Neural Network

For the actor network and the critic network, two neural networks with a three-layer network structure with the same number of parameters and structure shown in Figure 4 were prepared. Both neural networks use the LeakyReLU function as the activation function and are connected between the input and hidden layers and between the hidden and output layers.The input layer takes S in Section 3.3 as input in both networks. The difference between the actor network and the critic network is the output layer: the output layer of the actor network has $q = 1$, while the output layer of the critic network has $q = 2$.

3.5 Rewards

The reward function is a measure of the quality of one step of an action, and both the value function and the action value function are built on this basis. Therefore, the design of the reward function must be able to enable the control policy to avoid obstacles and fly to the destination; the PPO algorithm uses a clip function, and the size of one policy update is constrained by the hyperparameters. Therefore, to guarantee stable convergence and avoid local optima, the reward function should be continuous and differentiable. Based on these considerations, we designed a synthetic reward function with

Parameter Name	Value
Control period	0.02s
Max steps	2×10^8
Max step per episode	800
Dimension of states	423
Dimension of actions	2
Timesteps per batch	4056
Discount rate of reward	0.99
Parameter update times per iteration	20
Learning rate	3×10^{-3}
ϵ of clip function	0.2
Covariance matrix element value	0.6

Table 1: Hyperparameter

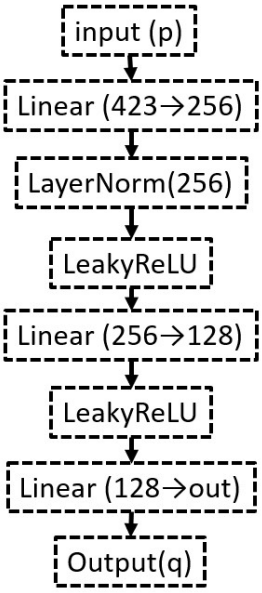


Figure 4: Network structure

the following five components and their weights.

$$R_{\text{total}} = R_{\text{dis}} + R_{\text{v}} + R_{\text{a}} + R_{\text{goal}} + R_{\text{ng}} \tag{2}$$

The first term R_{dis} is defined by Equation 3 based on the distance Dis_t between the UAV and the target location at the current step. This term rewards the UAV based on the amount of change in the distance between the UAV and the destination, and a positive reward is given when the UAV approaches the target.

$$R_{\text{dis}} = \begin{cases} k_{\text{dis}}(\text{Dis}_{t-1} - \text{Dis}_t), & \text{if } \text{Dis}_{t-1} - \text{Dis}_t > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

R_{v} is a constraint on the horizontal speed, denoted by Equation 4, which provides a negative reward if the horizontal speed of the UAV v_t exceeds $\text{speed}_{\text{max}}$. In this study, we limit it to 3m/sec.

$$R_{\text{v}} = \begin{cases} 0, & \text{if } v_t \leq \text{speed}_{\text{max}} \\ k_v(|v_t - \text{speed}_{\text{max}}|), & \text{if } v_t > \text{speed}_{\text{max}} \end{cases} \tag{4}$$

R_{a} is a penalty based on the UAV's velocity change, as expressed in Equation 5. It is defined as the sum of the absolute differences between the UAV's velocities in the X and Y axes at time t and those at time $t-1$. This term suppresses abrupt velocity changes and promotes smooth and stable motion, while contributing to reduced energy consumption and improved learning stability.

$$R_{\mathrm{a}} = k_a(|v_{x(t)} - v_{x(t-1)}| + |v_{y(t)} - v_{y(t-1)}|) \quad (5)$$

The R_{goal} is represented by Equation 6, which gives an additional positive reward if the target point is reached within a radius of 1 meter from the target point and a negative reward if the target point is moved away from after the target point is reached. This reward formulation is designed to encourage the agent to not only reach the target point but also to remain at the target location.

$$R_{\mathrm{goal}} = \begin{cases} k_{goal} & \text{if reach the destination} \\ k_{leave} & \text{if leave the destination} \end{cases} \quad (6)$$

R_{ng} provides a large negative reward as punishment if the UAV collides with an obstacle or reaches the maximum number of time steps in an episode without reaching the target point, as expressed in Equation 7. The episode is terminated when these conditions are reached in order to increase learning stability and efficiency.

$$R_{\mathrm{ng}} = \begin{cases} k_{obs} & \text{if collide with obstacle} \\ k_{step} & \text{if max step} \end{cases} \quad (7)$$

The weights of each reward function are as shown in Table 2.

4 EXPERIMENT

4.1 Experiment Environment

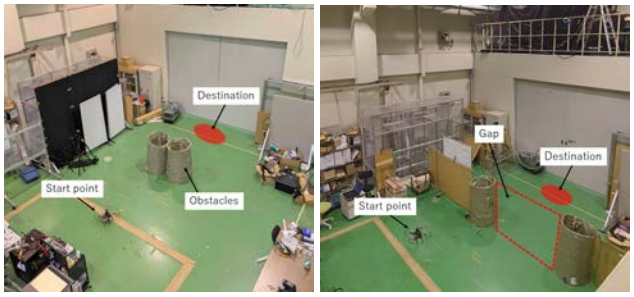
Experiments were conducted in an indoor environment, as shown in Figure 5. The UAV used in the experiment was the one shown in Figure 6. The size of the UAV is about $45\,\mathrm{cm} \times 45\,\mathrm{cm}$ and it is equipped with a LiDAR sensor and an on-board computer. The onboard computer was a Jetson Orin Nano from NVIDIA Corporation. Motion capture by OptiTrack was used instead of GNSS to acquire self-position.

4.2 Learning

In this study, since the validation experiments using the actual UAV were conducted indoors, we first examined the impact of environmental elements such as walls, excluding obstacles, on the learned policy. For this purpose, we constructed four types of learning environments that simulate indoor conditions. These included enclosed environments with

k_{dis}	2.5
k_v	0.1
k_a	0.01
k_{goal}	10
k_{leave}	-3
k_{obs}	-5
k_{step}	-3

Table 2: Weights of reward function



(a) env1 (b) env2∼env4

Figure 5: Experiment environment



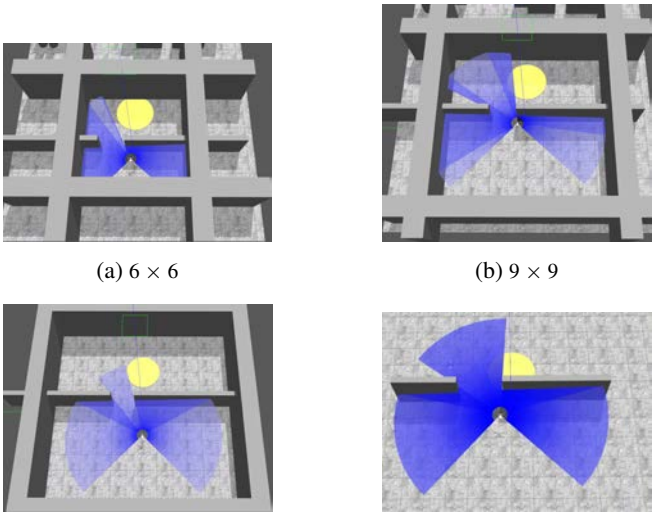
Figure 6: UAV for experiments

dimensions of 6m × 6m, 9m × 9m, and 12m × 12m, as well as an open environment without any walls, as illustrated in Figure 7. The position of the wall opening was fixed at $x = 1.0$ and $y = 0.5$ to 2.0. The effective utilization rates of the LiDAR sensor in each environment were approximately 100 percent in the 6m × 6m environment, between 70 and 90 percent in the 9m × 9m environment, and around 50 percent in the 12m × 12m environment, as shown in Figure 8. To assess the generalization performance, we conducted 100 trials under each training condition using the random environment shown in Figure 10. The success rates were 65 percent for the policy trained in the 6m × 6m environment, 33 percent for the 9m × 9m environment, and 11 percent for the 12m × 12m environment. In addition, the policy trained in the open environment was able to reach the target when tested under the same open conditions, but failed to do so in any of the indoor environments. These findings indicate that a higher utilization rate of LiDAR data during training is associated with improved generalization performance in unfamiliar environments.

Based on these results, for the verification experiment with the actual device, training was conducted in a Gazebo environment like that shown in Figure 9, where the wall distance, start position, target location, and gap position were

http://www.imavs.org/

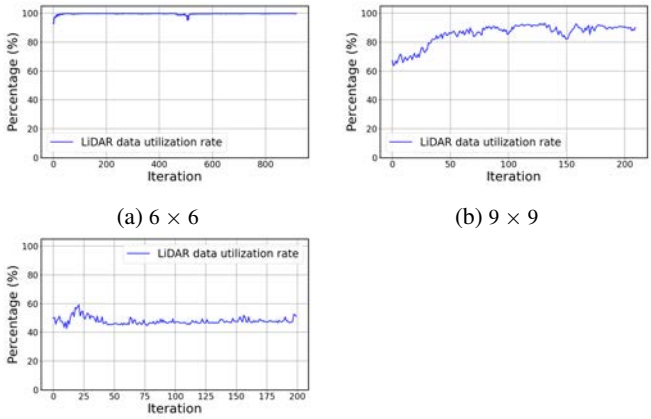
http://www.imavs.org/



(a) 6 × 6 (b) 9 × 9

(c) 12 × 12 (d) open space

Figure 7: Comparison by wall distance



(a) 6 × 6 (b) 9 × 9

(c) 12 × 12

Figure 8: Number of valid data for lidar data in the study

randomly varied as shown in Figure 10.

4.3 Result

To validate the performance of the proposed algorithm in a real-world environment, we designed the tasks of avoiding two cylinders and moving through three different gaps between two walls, as shown in Table 3. The UAV performs the tasks of avoiding obstacles or passing through gaps, reaching the destination, and hovering. The low-level controller runs on Pixhawk hardware, while the high-level control policy runs on a Jetson Orin Nano. The data nodes of the motion capture system run in ROS and communicate via a network interface. The UAV is then controlled to perform its mission in the real environment using the policies learned in the Figure 10 environment. An overview of the system used in this experiment is shown in Figure 11.

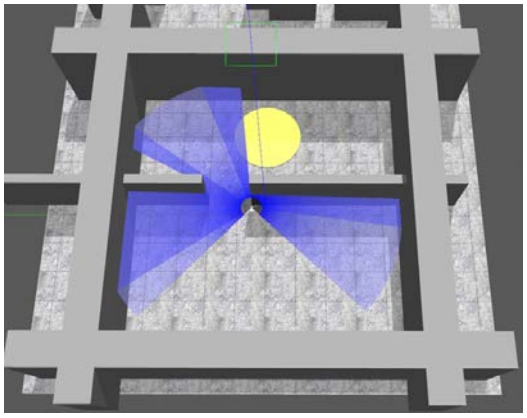


Figure 9: Simulation environment

The results of the position and velocity are shown in Figure 12 and Figure 13, respectively. The actual trajectories of the UAV are shown in Figure 14, and all experiments were conducted using the same policy. Results showed that the UAV was able to avoid obstacles and hover at the target point in all four environments using the same policy. This experiment demonstrates that in indoor environments, our method can control the UAV to complete the task of navigating complex real-world environments while sensing the environment using LiDAR sensors. In addition, by learning in a space surrounded by walls on all four sides with randomized wall distances and obstacle locations, we were able to obtain some generalization performance against obstacles, which was confirmed in an actual UAV. In particular, the experimental results of env1 showed that the robot was able to avoid obstacles with configurations and shapes that were not present in the learning process, confirming its generalization performance that can cope with unknown environments.

5 CONCLUSION

In this study, we applied deep reinforcement learning with end-to-end mapless navigation using LiDAR sensors in environments with unknown obstacles. While previous studies faced challenges in generalizing to new obstacle layouts,

	start	goal(r=1.0)	obs(r=0.3)	gap
env1	x = -2.0 y = 0	x = 2.0 y = 0	x = 0.25 y = 0, 0.6	\
env2	x = -1.5 y = -1.0	x = 2.5 y = 0	\	x = 0.5 y = -0.5 ∼ 1.0
env3	x = -1.5 y = 1.0	x = 2.5 y = 0	\	x = 0.5 y = -0.5 ∼ 1.0
env4	x = -1.5 y = 1.0	x = 2.0 y = 0	\	x = 0.25 y = -1.5 ∼ 0

Table 3: Experiment environment

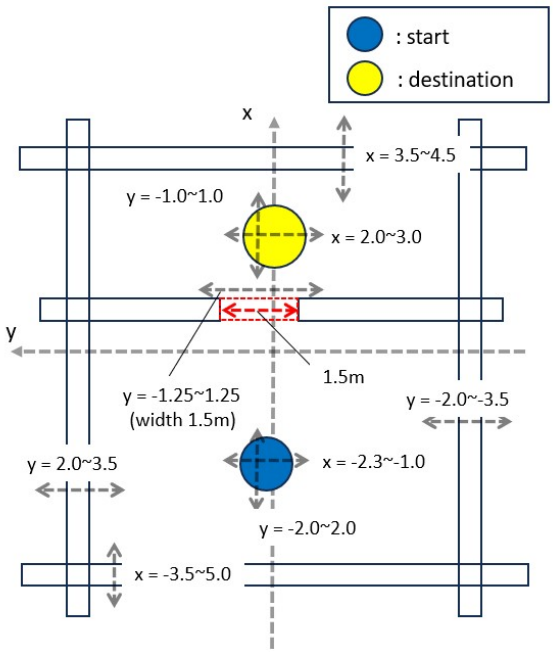


Figure 10: Randomized environment

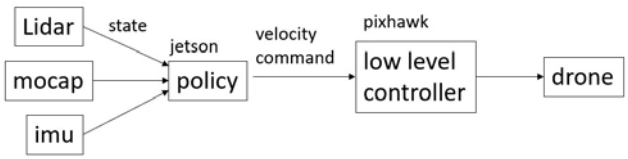
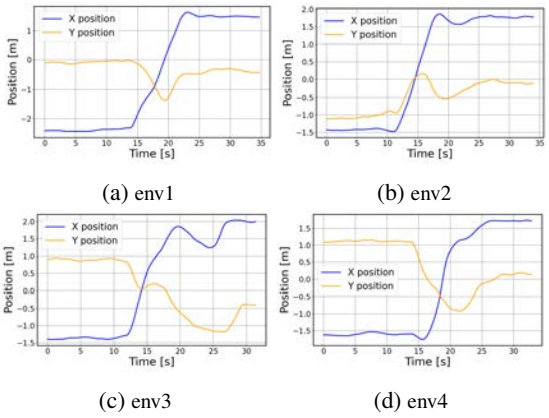


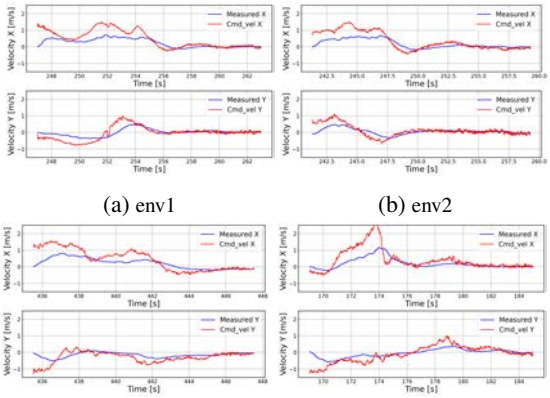
Figure 11: Real drone system



(a) env1 (b) env2

(c) env3 (d) env4

Figure 12: Position results for each environment



(a) env1 (b) env2

(c) env3 (d) env4

Figure 13: Speed results for each environment

our approach successfully achieved navigation adaptable to different obstacle placements, as confirmed through simulations and real-world experiments. We also investigated the impact of the effective data ratio from LiDAR on policy generalization. However, several issues remain. In real environments, elements such as luggage not present in simulations act as noise and degrade performance. Additionally, since LiDAR operates in two dimensions, high-speed navigation limits ground detection, reducing speed. The current experiments were based on simple obstacle layouts, so future work will explore more complex scenarios. Moreover, as the UAV currently relies on external sensors for localization, we plan to implement internal sensor-based estimation for onboard-only operation. Future evaluations will also include outdoor environments such as forests.

REFERENCES

[1] David C. Schedl, Indrajit Kurmi, and Oliver Bimber. An autonomous drone for search and rescue in forests using airborne optical sectioning. CoRR, abs/2105.04328, 2021.

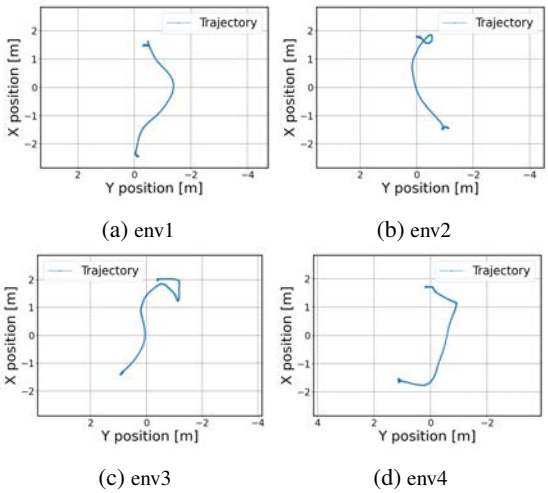
[2] Junwon Seo, Luis Duque, and Jim Wacker. Drone-enabled bridge inspection methodology and application. Automation in Construction, 94:112–126, 2018.

[3] Sebastien Boiteau, Fernando Vanegas Alvarez, Juan Sandino, Luis Gonzalez, and Julian Galvez. Autonomous uav navigation for target detection in visually degraded and gps denied environments. In Proceedings of the IEEE Aerospace Conference (AERO), pages 1–10, March 2023.

[4] S. A. H. Mohsan, N. Q. H. Othman, Y. Li, M. H. Alsharif, and M. A. Khan. Unmanned aerial vehicles (uavs): practical aspects, applications, open challenges, security issues, and future trends. Intelligent Service Robotics, 16(1):109–137, 2023.

[5] Benjamin Rivière, Wolfgang Hönig, Yisong Yue, and Soon-Jo Chung. Glas: Global-to-local safe autonomy synthesis for multi-robot motion planning with end-to-

http://www.imavs.org/



(a) env1 (b) env2

(c) env3 (d) env4

Figure 14: Trajectory results for each environment

end learning. IEEE Robotics and Automation Letters, 5(3):4249–4256, 2020.

[6] Yiming Mao, Ming Chen, Xianglin Wei, and Bing Chen. Obstacle recognition and avoidance for uavs under resource-constrained environments. IEEE Access, PP:1–1, 08 2020.

[7] Yiming Mao, Ming Chen, Xianglin Wei, and Bing Chen. Obstacle recognition and avoidance for uavs under resource-constrained environments. IEEE Access, 8:169408–169422, 2020.

[8] Hongxun Liu and Satoshi Suzuki. Model-free guidance method for drones in complex environments using direct policy exploration and optimization. Drones, 7(8), 2023.

[9] Christos Chronis, Georgios Anagnostopoulos, Elena Politi, George Dimitrakopoulos, and Iraklis Varlamis. Dynamic navigation in unconstrained environments using reinforcement learning algorithms. IEEE Access, 11:117984–118001, 2023.

[10] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. CoRR, abs/1707.06347, 2017.

[11] Tengteng Zhang and Hongwei Mo. Reinforcement learning for robot research: A comprehensive review and open issues. International Journal of Advanced Robotic Systems, 18(3), 2021.

[12] Fadri Furrer, Michael Burri, Markus Achtelik, and Roland Siegwart. RotorS—A Modular Gazebo MAV Simulator Framework, pages 595–625. Springer International Publishing, Cham, 2016.

http://www.imavs.org/

Radio Coverage-Aware UAV Planning and Navigation in Urban Scenarios

D. Aláez*, M. Prieto, J. Villadangos, J.J. Astrain
Statistics, Computer Science and Mathematics Department,
Universidad Pública de Navarra, Pamplona 31006, Spain

http://www.imavs.org/

ABSTRACT

We propose a novel UAV path planning and navigation solution for urban environments, integrating radio path loss into a modified A* path planning algorithm. The proposed algorithm is based on a fast raytracing algorithm, which enables any 3D scenario to be modeled for specific radio link conditions. The radio coverage map is then fed along with an obstacle occupancy grid to the algorithm to find the optimal path that avoids obstacles and minimizes path loss. Additionally, we introduce a smoothing step to generate UAV-suitable paths while maintaining radio performance. To verify the implementation, a test case environment has been proposed in the campus of the Public University of Navarra. According to the proposed test cases, the overall path loss is reduced up to 24.12% by using our methodology.

1 INTRODUCTION AND RELATED WORKS

Autonomous aerial mobility in densely built environments is rapidly transforming urban transportation, providing novel approaches to alleviating traffic congestion and improving logistical efficiency. Integrating Unmanned Aerial Vehicles (UAVs) into urban airspace offers significant potential for accelerating deliveries, transporting medical supplies, and enhancing emergency response capabilities.

Effective UAV navigation in urban scenarios presents significant challenges due to the presence of obstacles such as buildings and the need to maintain reliable radio communication. Ensuring optimal path planning that balances obstacle avoidance and communication quality is critical, especially in communication-sensitive tasks. According to [1], nearly a third of civil UAV accidents experienced communication loss.

A growing body of research examines the integration of collision avoidance strategies within path planning, ensuring safety during UAV operations. López et al. propose collision risk management strategies for multi-UAV systems, where autonomous navigation is critical to avoid obstacles and maintain airspace safety [2]. The utilization of artificial intelligence (AI) in path planning can also enhance the responsiveness of UAVs to dynamic urban environments through real-time adjustments and risk-aware strategies, as demonstrated in [3]. These advancements are increasingly essential as the complexity of urban environments demands sophisticated algorithms capable of adjusting to sudden changes during flight.

The safety of urban air mobility is further compromised by the risk associated with third-party damage. Path planning must account for third-party risk while ensuring compliance with regulations governing UAV operations, particularly in urban areas crowded with obstacles. Path planning methodologies that consider the avoidance of high-risk zones are gaining attention as effective strategies for mitigating these risks; Tang et al. introduce a risk modeling approach to inform UAV flight paths in urban settings [4].

Traditional path planning algorithms such as Dijkstra's [5], A* [6], and RRT* [7] prioritize spatial optimization, often neglecting radio coverage considerations, which can lead to signal degradation in dense urban areas. A systematic review on recent advances in autonomous mobile robot navigation can be found in [8]. Other authors propose radio map-based UAV path planning [9, 10, 11], but rely on statistical models or only consider topographic data.

To address this gap, we propose a novel modification of the A* algorithm that incorporates radio propagation data, derived from a fast raytracing simulator based on a Multipath Radio Tracer (MURT [12]) for 3D urban photogrammetry data. By integrating path loss (PL) values directly into the cost function of the A* algorithm, our approach generates paths that not only avoid obstacles but also maximize signal strength. In addition, a smoothing step is applied to enhance the trajectory of the UAV while respecting the radio coverage requirements. This method ensures efficient UAV navigation, making it highly suitable for applications such as UAV-assisted wireless networks and urban emergency operations. A simplified overview of the proposed contributions, organized by steps is shown in Figure 1.

2 MATERIALS AND METHODS

Let the environment be represented by a discrete voxel grid $G(i, j, k)$, where each voxel represents a position in the 3D space. Let path loss $PL(i, j, k)$ be a scalar value associated with each voxel. The goal is to find the path that minimizes the overall PL between start location $s = (x_s, y_s, z_s)$ and final location $e = (x_e, y_e, z_e)$.

*Email address(es): contact_author@mail.com

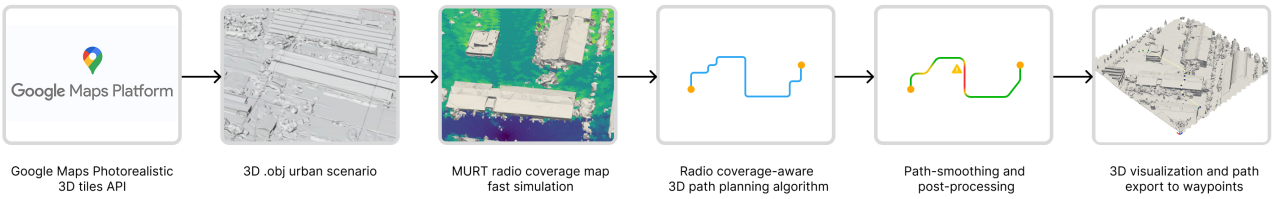


Figure 1: Step by step process diagram for obtaining a radio coverage-aware path plan for UAV flight in dense urban scenarios.

http://www.imavs.org/

PL is the attenuation of the signal as it propagates through space and is affected by: (a) grid location: the position of the voxel (i, j, k); (b) transmitter location: the position of the transmitter (x_{tx}, y_{tx}, z_{tx}) as the source of the radio control signal; (c) frequency: the frequency of the signal impacts how it interacts with materials and how it propagates through space; (d) material properties: different material properties cause different levels of absorption, reflection, and diffraction of the signal, based on their electromagnetic properties (dielectric constant, permeability, conductivity); (e) obstacles: obstacles between transmitter and receiver (e.g. walls, buildings) affect the direct line-of-sight path, leading to additional losses due to shadowing effects and multipath; (f) interferences caused by other emitting sources such as other aircrafts.

2.1 *Obtaining a PL model for a given scenario*

Using Google's Photorealistic 3D tiles API[1], urban scenarios can be downloaded for a set of coordinates as `.obj` files that may be directly imported into the simulator. The mesh is then voxelized to calculate the number of points to be simulated using trimesh [13]. This voxelized map facilitates obtaining the binary occupancy grid that will later be used to search for optimal collision-free paths. For fast computation of PL in all voxels, a modified version of MURT is used [12]. MURT is a Python ray-tracing engine for multipath propagation of radio waves, with the ray-tracing core engine implemented in Python C++ Extension. This ray tracing algorithm computes the PL for the most common propagation mechanisms: direct line-of-sight, reflection, and diffraction [14].

The **direct PL** calculation is based on the Friis transmission equation[15], and is used to compute the free-space path loss (FSPL) over a direct line-of-sight path between the transmitter (x_{tx}, y_{tx}, z_{tx}) and the receiver (x_{rx}, y_{rx}, z_{rx}):

$$PL_{LOS}(d, f) = 20\log_{10}(d) + 20\log_{10}(f) - 20\log_{10}\left(\frac{4\pi}{c}\right), \tag{1}$$

where d is the Euclidean distance between the transmitter and the receiver, f is the transmission frequency in Hz, and c is the speed of light in m/s.

[1]Google Photorealistic 3D Tiles official website: `https://developers.google.com/maps/documentation/tile/3d-tiles`

The **reflection PL** accounts for the reflection of the signal off a surface, using Fresnel reflection coefficients. The reflection coefficient depends on the angle of incidende computed using Snell's law: $n_1 \sin\theta_1 = n_2 \sin\theta_2$, where n_1 and n_2 are the refractive indices of air and the material, and θ_1 and θ_2 are the angles of incidence and transmission, respectively. The reflection coefficient also depends on whether the wave is transverse electric (TE) or transverse magnetic (TM) polarized:

$$R_{\parallel} = \frac{n_1\cos\theta_1 - n_2\cos\theta_2}{n_1\cos\theta_1 + n_2\cos\theta_2}, \quad R_{\perp} = \frac{n_2\cos\theta_1 - n_1\cos\theta_2}{n_2\cos\theta_1 + n_1\cos\theta_2} \tag{2}$$

The reflection coefficient is used to adjust the PL equation:

$$PL_r(d_{tot}, f, R) = 20\log_{10}(d_{tot}) + 20\log_{10}(f) - 20\log_{10}(|R|) - 20\log_{10}\left(\frac{4\pi}{c}\right), \tag{3}$$

where d_{tot} is the total distance traveled by the reflected path.

The **diffraction PL** is computed using the knife-edge diffraction model, which approximates diffraction around obstacles like edges. The diffraction parameter ν is computed as:

$$\nu = h\sqrt{\frac{2d_{tx-rx}}{\lambda d_{tx-edge} d_{edge-rx}}}, \tag{4}$$

where h is the effective height of the obstacle relative to the LOS path and λ is the wavelength of the transmitted signal. The diffraction loss uses the Fresnel integral approximation:

$$PL_d(\nu) = 6.9 + 20\log_{10}\left(\sqrt{(\nu - 0.1)^2 + 1} + \nu - 0.1\right). \tag{5}$$

Lastly, the total PL is computed considering the contributions of multiple propagation paths and summing their effects in the linear domain:

$$PL_{total} = -10\log_{10}\left(\sum_{i=1}^{n} 10^{-\frac{PL_i}{10}}\right) \tag{6}$$

Summing PL directly (even in the linear domain) is an oversimplification that ignores the phase-dependent nature of

multipath propagation (particularly constructive and destructive interference effects). Averaging out multipath interference effects over time when dealing with large distances or in highly scattered environments is far less precise than conventional raytracing simulators and neglects detailed fading patterns [16]. However, it allows simulation times to be significantly reduced to almost real time. Studies are currently being conducted to compare the applicability and limitations of this algorithm against a proven ray tracing simulator [17].

2.2 Finding the optimal path

Given a path P, which is a sequence of voxels $P = \{p_0, p_1, ..., p_n\}$, where $p_0 = s$ (start) and $p_n = e$ (end), the total PL can be defined as the sum of PL through all the nodes in the path:

$$\mathcal{L}(P) = \sum_{i=0}^{i=n} PL(p_i), \text{ s.t. } p_0 = s,\ p_n = e \tag{7}$$

This is the cost function which we mainly aim to minimize. The problem is subject to two constraints:

1. Occupancy constraint: the path can only pass through free spaces, i.e., voxels that are not occupied by obstacles. This is typically represented by a binary occupancy grid:

$$G(p_i) = 0, \forall i = 0, 1, ..., n \tag{8}$$

2. Movement constraint: The movement between adjacent voxels should follow one of the valid moves (e.g., six connected neighbors in a 3D space).

Each voxel of the binary occupancy grid can be associated to a scalar value representing the path loss for a given radio setup.

There are numerous path finding algorithms in the literature [18], but the classical A* algorithm [6] generally provides an intuitive well-performing solution to this problem. The A* algorithm is a heuristic search algorithm that finds the path with the lowest cost between two points. Each node p_i is associated with a cost function $f(p_i)$, which is the sum of two components:

1. Cost of travelling from the start node s to the current node p_i: $g(p_i)$.
2. Heuristic: an estimate of the remaining cost to reach the goal node e from the current node p_i, denoted as $h(p_i)$. A common choice is the Manhattan distance: $h(p_i) = |x_i - x_e| + |y_i - y_e| + |z_i - z_e|$

Thus, the total cost function at node p_i is given by:

$$f(p_i) = g(p_i) + h(p_i), \tag{9}$$

where $g(p_i)$ represents a combination of the path loss (PL) and the distance traveled from the start node to the current node, and $h(p_i)$ is the estimated cost to reach the goal node. We have opted for modifying the conventional step cost $g(x)$ to consider a weighted combination of distance and PL:

$$g(x_{i+1}) = g(x_i) + (1 - \lambda) - \lambda PL_{norm} \tag{10}$$

where $\lambda \in [0, 1]$ is the coverage-over-distance weight parameter, and PL_{norm} is the normalized path loss reward, where $PL_{norm} = 1$ means best path loss and $PL_{norm} = 0$ is the worst-possible path loss value.

This is a classical graph problem of heuristic search. Dealing with multiple optimization criteria consists of finding paths that are optimal according to more than one attribute. A common approach to include multi-criteria into the search is to define edge costs $\sum_i \lambda_i w_i$, with $\sum_i \lambda_i = 1$. Albeit being a popular approach in videogames, the optimization cost algebra must be generalized to ensure the admissibility of the exploration algorithm, i.e., the fact that it solves the optimality problem. As depicted in [19], a heuristic function $h : V \to A$ with $h(t) = \mathbf{1}$ for each goal node $t \in T$ is

- admissible, if for all $u \in V$ we have $h(u) \preceq \delta(u, T)$, i.e., h is a lower bound, denoting $\delta(u, V)$ as the cost of the optimal path starting at node u and reaching node v in a set V.
- consistent, if for each $u, v \in V$ and $e \in E$, such that $u \xrightarrow{e} v$, we have $h(u) \preceq \omega(e) \times h(v)$.

Therefore, the coverage-over-distance weight parameter λ should be selected such that both conditions are met to ensure admissibility of the optimal path planning algorithm.

The algorithm uses an open list to add nodes from the start node s up to the goal node e by selecting nodes with the lowest $f(p_i)$. The path is then reconstructed by following the parent pointers of each node.

The generated path based on contiguous movements in a grid differs from the usual waypoint-based navigation employed with UAV. To reduce unnecessary waypoints and allow movement in any direction, a path smoothing algorithm is used. The path smoothing algorithm works by iterating over the given path and trying to remove unnecessary intermediate points. For each point p_i, it checks if the direct line between p_i and a later point p_j is obstacle-free and that the deviation of intermediate points between p_i and p_j from the straight line is less than a specified maximum deviation d_{max}. The pseudocode is detailed in Algorithm 1.

http://www.imavs.org/

Algorithm 1 Path Smoothing Algorithm

Require: Path P, grid G, maximum deviation $\delta_{\max}$, buffer $b \leftarrow 0$
Ensure: Smoothed path S

$S \leftarrow [P_0]$
$i \leftarrow 0$
while $i < |P| - 1$ **do**
  $j \leftarrow i + 1$
  while $j < |P|$ **do**
    if LINEOFSIGHT(G, P_i, P_j, b) **then**
      if $j - i > 1$ **then**
        $\delta \leftarrow \max_{k=i+1}^{j-1}$ DISTFROMLINE(P_k, P_i, P_j)
        if $\delta > \delta_{\max}$ **then**
          break
        end if
      end if
      $j \leftarrow j + 1$
    else
      break
    end if
  end while
  Append P_{j-1} to S
  $i \leftarrow j - 1$
end while
return S

The straight line is discretized using Bresenham's 3D Line algorithm [20], obtaining all grid points that lie approximately along the straight line between two selected nodes. The differences in each axis are calculated as follows: $\Delta x = |x_2 - x_1|, \Delta y = |y_2 - y_1|, \Delta z = |z_2 - z_1|$ The dominant axis is incremented step by step, while the other axes are incremented according to a continuous error term to approximate the straight line. Line-of-sight check verifies that for all points $p_i(x_i, y_i, z_i)$ along the Bresenham-generated line, the grid value at that point is: $G_i(x_i, y_i, z_i) \neq 1$ (obstacle). Lastly, the distance of a point from a line segment is used to ensure that the new path does not differ from the optimal PL path, so that the PL estimations remain valid through the smoothed path.

2.3 Calculating conflicting path segments for enhanced decision making

Given a PL-optimal smoothed path, the PL of each voxel the path goes through can be obtained thanks to the Bresenham algorithm. The PL for the maximum range can be expressed as:

$$PL = G_{Tx} + G_{Rx} + P_{Tx} - S_{Rx} - LM, \qquad (11)$$

where G_{Tx} is the transmitter gain, G_{Rx} the receiver gain, P_{Tx} the transmitter power, S_{Rx} the receiver sensitivity, and LM the link margin. Since the link margin, transmitter gain, and receiver gain are typically fixed during flight, the transmitter power and receiver sensitivity can be dynamically adjusted at specific locations where PL is expected to exceed the threshold that allows direct communication with the aircraft. By having this information before the flight, the transmission and reception power needs can be intelligently planned to adapt it dynamically during the operation, avoiding saturating the radio channel in urban environments and increasing power efficiency.

3 RESULTS AND DISCUSSION

The proposed system has been implemented in Python and is publicly accessible from the Github repository (`https://github.com/UPNAdrone/UrbanPathPlanner`). The PL at each position is obtained from the ray launching simulation and converted into a point cloud matching the voxel grid coordinates. The binary occupancy grid is also extracted from the ray launching simulation setup. The voxel mesh is generated using `trimesh`. Lastly, `pyvista` is used for interactive 3D visualization of the original mesh and an interactive slice map of the radio coverage for the selected height.

To verify the implementation, a test case environment has been proposed in the campus of the Public University of Navarra. Following the steps described in the previous Sections, the coverage map for a 868 MHz transmitter at different locations has been obtained, and the optimal path has been estimated for different paths between different locations of the campus. Four randomized transmitter locations are selected, along with two random initial and final locations from a pool of flyable take-off and landing zones. The results are shown in Table 4. As can be seen, the overall PL is reduced from 6.21% up to 24.12% by using our methodology.

For reference, in Figure 2 the path selected by A* using the conventional cost function where only distance is considered is shown on the left, the path selected using our proposed cost function (considering PL) is shown on the middle, and the coverage map for each experiment is shown on the right.

4 CONCLUSION

Throughout this paper, we have proposed a novel UAV path planning and navigation solution for urban environments, integrating radio PL into a modified A* path planning algorithm, including a fast raytracing algorithm, a binary occupancy grid, and a path-smoothing algorithm. As a complement to this work, we propose using the expected coverage information at each point of the path to adjust the transmitter and receiver power in complex urban scenarios in real time, providing the aircraft with greater intelligence and autonomy. Although this work is based on a modification of A*, the algorithm is currently being generalized to any heuristics-based path planning algorithm.

For future works, we propose using real-time planning algorithms that allow the flight path to be adapted based on changing conditions and new information that may be avail-

http://www.imavs.org/

http://www.imavs.org/

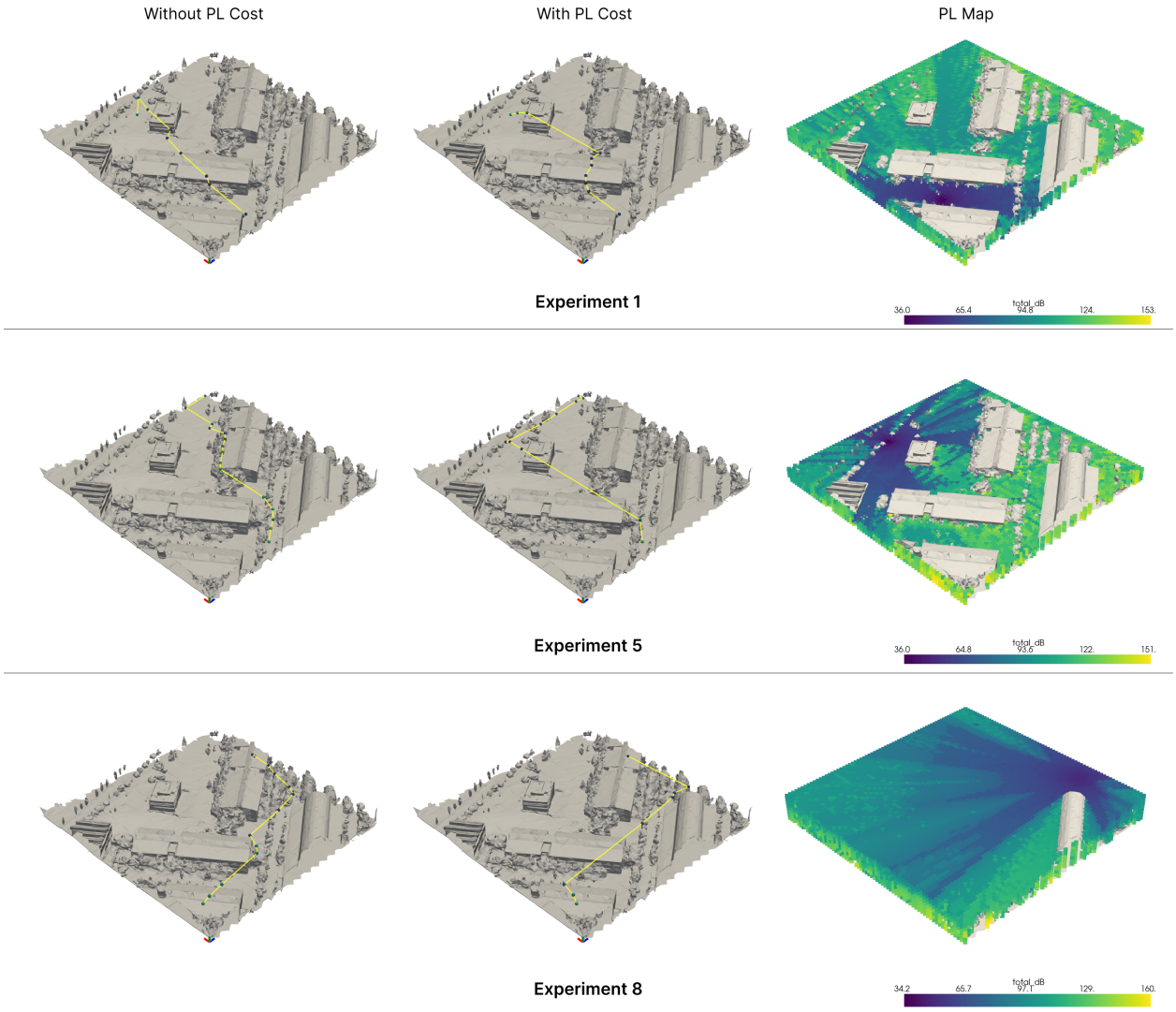


Figure 2: Experiments 1, 5 and 8. On the left: path generated by A* without considering PL. On the middle: path generated considering PL. On the right: PL map slice for reference.

able. Finally, we propose replacing the simplified ray tracing algorithm with one that considers aspects such as different materials, interference effects for multipath, and other complex phenomena.

References

[1] Graham Wild, John Murray, and Glenn Baxter. Exploring civil drone accidents and incidents to help prevent potential air disasters. *Aerospace*, (3), 2016.

[2] B. M. López, J. Muñoz, F. Quevedo, C. A. Monje, S. Garrido, and L. Moreno. Path planning and collision risk management strategy for multi-uav systems in 3d environments. *Sensors*, 21:4414, 2021.

[3] S. Primatesta, G. Guglieri, and A. Rizzo. A risk-aware path planning strategy for uavs in urban environments. *Journal of Intelligent Amp; Robotic Systems*, 95:629–643, 2018.

[4] H. Tang, Q. Zhu, B. Qin, R. Song, and Z. Li. Uav path planning based on third-party risk modeling. *Scientific Reports*, 13, 2023.

[5] Edsger W Dijkstra. A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: his life, work, and legacy*, pages 287–290. 2022.

[6] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum

Table 1: Path planning algorithm comparison between standard A* and our modified version including PL.

ID	Tx Position [m]	Start [m]	End [m]	Total PL (standard)	Total PL (ours)	% Variation
1	49, 10, 26	53, 2, 25	12, 6, 45	5293.30	4964.52	-6.21
2	49, 10, 26	53, 2, 25	15, 6, 174	9016.10	7605.66	-15.64
3	70, 12, 80	7, 3, 21	162, 9, 12	5846.50	4805.77	-17.80
4	70, 12, 80	21, 2, 40	156, 6, 165	5904.62	4571.16	-22.58
5	170, 8, 80	3, 2, 23	186, 6, 180	9868.78	7766.59	-21.30
6	170, 8, 80	60, 3, 2	54, 12, 183	9885.72	7500.93	-24.12
7	50, 13, 180	56, 4, 12	33, 9, 102	6752.15	6078.43	-9.98
8	50, 13, 180	6,5,4	117, 18, 171	7002.20	6224.48	-11.11

http://www.imavs.org/

cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[7] Sertac Karaman, Matthew R. Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the rrt*. In *2011 IEEE International Conference on Robotics and Automation*, pages 1478–1483, 2011.

[8] Anbalagan Loganathan and Nur Syazreen Ahmad. A systematic review on recent advances in autonomous mobile robot navigation. *Engineering Science and Technology, an International Journal*, 40:101343, 2023.

[9] Shuowen Zhang and Rui Zhang. Radio map-based 3d path planning for cellular-connected uav. *IEEE Transactions on Wireless Communications*, 20(3):1975–1989, 2021.

[10] Behzad Khamidehi and Elvino S. Sousa. Federated learning for cellular-connected uavs: Radio mapping and path planning. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pages 1–6, 2020.

[11] Yangrui Dong, Chen He, Ziteng Wang, and Lin Zhang. Radio map assisted path planning for uav anti-jamming communications. *IEEE Signal Processing Letters*, 29:607–611, 2022.

[12] Supawat Tamsri. Murt, 4 2021.

[13] Dawson-Haggerty et al. trimesh, 12 2019.

[14] Franco Fuschini, Hassan El-Sallabi, Vittorio Degli-Esposti, Lasse Vuokko, Doriana Guiducci, and Pertti Vainikainen. Analysis of multipath propagation in urban environment through multidimensional measurements and advanced ray tracing simulation. *IEEE Transactions on Antennas and Propagation*, 56(3):848–857, 2008.

[15] Harald T Friis. A note on a simple transmission formula. *Proceedings of the IRE*, 34(5):254–256, 1946.

[16] Bernard Tamba Sandouno, Yamen Alsaba, Chadi Barakat, Walid Dabbous, and Thierry Turletti. A novel approach for ray tracing optimization in wireless communication. *Computer Communications*, 209:309–319, 2023.

[17] L Azpilicueta, Meenakshi Rawat, Karun Rawat, F Ghannouchi, and F Falcone. Convergence analysis in deterministic 3d ray launching radio channel estimation in complex environments. *The Applied Computational Electromagnetics Society Journal (ACES)*, pages 256–271, 2014.

[18] Azad Noori and Farzad Moradi. Simulation and comparison of efficency in pathfinding algorithms in games. *Ciência e Natura*, 37(6-2):230–238, 2015.

[19] Stefan Edelkamp, Shahid Jabbar, and Alberto Lluch-Lafuente. Cost-algebraic heuristic search. In *AAAI*, volume 5, pages 1362–1367, 2005.

[20] J. E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Syst. J.*, 4(1):25–30, March 1965.

ADRC-Driven Trajectory Control of a Quadcopter Considering Ground Effect Dynamics

Carlos Osorio Quero*, Jose Martinez-Carranza
Computer Science Department,Instituto Nacional de Astrofisica Optica y Electronica,
Luis Enrique Erro #1, Mexico,Puebla,72810.

http://www.imavs.org/

Abstract

The Active Disturbance Rejection Control (ADRC) strategy is designed to mitigate the destabilizing effects of proximity to the ground, commonly known as the ground effect, on quadcopter drones during low-altitude flight. The ground effect introduces complex aerodynamic interactions that compromise flight stability and control precision. The proposed ADRC framework combines feedback linearization with real-time disturbance estimation to address this, enabling robust trajectory tracking despite nonlinear and variable disturbances. High-fidelity Computational Fluid Dynamics (CFD) simulations were used to model ground-effect dynamics and inform controller design. Unlike conventional controllers that rely heavily on computationally intensive CFD models during operation, the ADRC approach adaptively compensates for disturbances, enhancing real-time reliability. The simulation results validate the controller's capacity to maintain accurate altitude and stable flight in ground-effect conditions. These findings highlight the potential of ADRC-based methods to improve the robustness and maneuverability of control in near-ground drone operations.

1 Introduction

Small quadcopter UAVs are increasingly being used for applications such as infrastructure inspection [1], delivery [2], precision agriculture [3], pollution monitoring [4], and search and rescue missions [5]. These tasks often require operation in confined environments and proximity to surfaces, which introduces significant aerodynamic challenges, notably the ground effect (GE) [6].

GE arises from altered airflow patterns near rigid surfaces such as the ground or ceiling, affecting the efficiency of the rotor, reducing the induced velocity, and modifying the aerodynamic drag. Although this effect has been extensively modeled in helicopters and single-rotor systems, multirotor UAVs present greater complexity due to rotor interactions, making accurate GE modeling more difficult.

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

Although various studies have focused on the characterization of GE, mathematical modeling, and computational fluid dynamics (CFD) simulations [7], most UAV control strategies neglect or avoid GE by maintaining altitude. However, emerging research emphasizes the need for control systems that can actively compensate for these disturbances, especially during low-altitude maneuvers. One such approach is Active Disturbance Rejection Control (ADRC) [8, 9], which offers a model-independent method for the estimation and rejection in real-time of external disturbances such as GE.

In this work, we use CFD simulations (via SolidWorks Flow Simulation) to create high-fidelity aerodynamic data sets under different ground proximities (1 m, 40 cm, 20 cm, 10 cm) with UAVs equipped with 5-inch propellers rotating at 5000–8000 rpm. These data sets validate an ADRC-based trajectory tracking system designed to improve flight stability and accuracy in the presence of strong GE. The proposed framework demonstrates improved performance in scenarios that are critical for UAVs operating in realistic missions near the ground. Therefore, in this work, we propose the following:

- Analyzing the Computational Fluid Dynamics (CFD) of UAVs within and beyond ground effect zones (GE).
- Implementation of ADRC-ESO and RBFNN ADRC-GESO approaches to address perturbations induced by ground effects in UAVs.

2 Related Works

The study of the effect of ground in rotorcraft has progressed from its origins in helicopter aerodynamics to applications in small UAVs. Early models, such as the one by Cheeseman and Bennett [10] using momentum theory and image methods, laid the foundation by treating the rotor as a point source to predict thrust enhancement and power reduction near the ground. Although widely adopted for UAV control, these quasi-steady models [11], often dependent on normalized rotor height, struggle to generalize between different UAV configurations due to their sensitivity to rotor height, blade geometry, pitch angle, and other design factors.

More recent efforts have expanded to multirotor systems [12] with empirical corrections, yet such models remain limited by singularities near the ground and a dependence on UAV-specific data. To overcome these limitations, researchers are now exploring adaptive, data-driven methods,

http://www.imavs.org/

such as machine learning and real-time state estimation, for broader applicability.

Experimental studies using tools such as PIV [13] and force measurements have investigated the behavior of the rotor near partially ground or ceiling surfaces during forward flight, but have not yet produced fast and reliable empirical models. In response, the authors previously introduced an improved quasi-steady model [14] based on the theory of blade elements, which incorporates the pitch, solidity, and rotor geometry of the blade. This model accurately predicts finite IGE thrust [15] and considers dynamic effects such as forward speed and advance ratio, improving thrust estimation and offering practical benefits for UAV control and rotor design, as defined in Eq.(1)[16].

$$\mu = \frac{V_\infty}{\|\Omega\| R} \tag{1}$$

V_∞ denotes the airspeed in the free stream, which is usually equivalent to the vehicle's true airspeed, and $\|\Omega\|$ represents the rotor's rotational speed. Experimental observations have shown that the advance ratio μ significantly influences the in-ground effect (IGE). However, as Cheeseman and Bennett noted (see Table 1), the thrust enhancement provided by IGE diminishes and can be completely negated when $\mu > 0.1$ [17]. In their work [18], they derived an analytical expression for IGE under forward flight conditions, as presented in Eq. (2) [19].

$$K_{IGE} \equiv \frac{T(z)}{T_\infty} = \frac{1}{1 - \frac{1}{16}\left(\frac{R}{Z}\right)^2 / \left(1 + \left(\frac{V}{v_i}\right)^2\right)} \tag{2}$$

The interaction between a rotor and the ground, known as the in-ground effect (IGE), significantly alters thrust generation during flight. In this context, V represents the vehicle velocity, while v_i denotes the rotor-induced velocity. The effect of the ground during forward flight has been modeled using a state-space approach [20, 21], predicting a monotonic decrease in the thrust ratio with increasing advance ratio $/mu$. However, the experimental results [22] reveal nonmonotonic behavior, including a recirculation region for $0.04 \leq \mu \leq 0.06$ and rotor heights z between 0.6 and 1.2, which reduces rotor thrust. At higher μ values, the ground vortex reintroduces IGE effects. Particle Image Velocimetry (PIV) studies [23] further highlight critical flow characteristics, such as fountain and ground vortices, which correlate with increased power consumption and decreased thrust.

3 ACTIVE DISTURBANCE REJECTION CONTROL WITH A GROUND-EFFECT MODEL

We use an 8-state quadrotor model for translation and rotation, $x \triangleq \begin{bmatrix} z & v_z & \phi & p & \theta & q & \psi & r \end{bmatrix}$ and $u \triangleq \begin{bmatrix} T_{\text{cmd}} & \tau_\phi & \tau_\theta & \tau_\psi \end{bmatrix}$ with known constants g, m, and principal inertias I_x, I_y, I_z. A 12-state ESO (three observer states per controlled channel $i \in \{z, \phi, \theta, \psi\}$) estimates $\begin{bmatrix} \hat{x}_i, & \hat{d}_i, & \widehat{\dot{d}_i} \end{bmatrix}$, where $\hat{d}_i$ lumps unmodeled dynamics and disturbances. To capture near-ground aerodynamics, the altitude channel introduces a height-dependent thrust multiplier $\gamma(z) \geq 1$ (ground effect). The altitude subsystem of x, $x_z = [z \;\; v_z]^\top$, evolves as

$$\dot{z} = v_z, \qquad \dot{v}_z = \frac{T_{\text{cmd}}\, \gamma(z)}{m} - g, \tag{3}$$

Under the small-angle assumption (the inner attitude loop keeps ϕ, θ small, so that the vertical thrust component matches the commanded thrust). The remaining attitude states follow the standard rigid-body rotational dynamics driven by $(\tau_\phi, \tau_\theta, \tau_\psi)$. The ESO uses the measured outputs $y = [z \quad \phi \quad \theta \quad \psi]^\top$ to update the channel disturbance estimates, with $\hat{d}_z$ absorbing residual model error in (3) and the ground-effect correction through $\gamma(z)$. T_{cmd} is the total thrust commanded by the controller, and $\gamma(z)$ is a ground effect factor that amplifies thrust if the UAV is sufficiently close to the ground. Rotational channels, roll, pitch, and yaw, each follow the basic form of the Euler equations with a rigid body.

3.1 *Generalized Extended State Observer (GESO) in the Altitude Loop*

We modeled the ground effect with a dimensionless thrust multiplier $\gamma_g(h) \geq 1$, and estimate it online as $\hat{\gamma}_g$ within a generalized extended state observer (GESO). The observer updates $\hat{\gamma}_g$ from the altitude innovation (measured minus predicted). A CFD-derived ground effect map provides $\gamma_g(h)$ via piecewise linear interpolation: it is about 2.2 at $h = 0.10\,\text{m}$ and tapers to 1.0 for $h \geq 1\,\text{m}$. Incorporating $\hat{\gamma}_g$ into the vertical control produces height-dependent thrust compensation and improves the low-altitude tracking accuracy. The corrected thrust command is

$$T_{\text{cmd}} \;=\; \frac{m\big(g + a_z^{\text{cmd}} - \hat{d}_z\big)}{\text{clip}\big(\hat{\gamma}_g,\, 1,\, \gamma_{\max}\big)}\,, \tag{4}$$

where $\text{clip}(x, 1, \gamma_{\max}) = \min\{\max\{x, 1\}, \gamma_{\max}\}$ ensures physical bounds in the multiplier estimate.

3.2 *Control Strategy with Active Disturbance Rejection and Ground-Effect Compensation*

Operating multi-rotor UAVs near the ground introduces complex, non-linear aerodynamic disturbances that can destabilize flight. To address this, the proposed control strategy uses radial basis function neural networks (RBFNN) to adaptively tune the gains of the controller. This approach dynamically adjusts the proportional and derivative gains in the roll, pitch and yaw loops based on real-time error states, improving stability and robustness under varying ground-effect conditions Eq. (5).

$$\left(k_{p_\phi}, k_{d_\phi}, k_{p_\theta}, k_{d_\theta}, k_{p_\psi}, k_{d_\psi}\right) = RBFNN\left(e_\phi, p, e_\theta, q, e_\psi, r\right) \tag{5}$$

Table 1: Compares analytical models used to estimate thrust variations under in-ground effect (IGE) conditions. Each model is assessed based on its applicability to multi-rotor platforms and forward flight scenarios, T aerodynamic thrust produced by the rotor system, T_{IGE} thrust generated in ground effect , T_{OGE} thrust generated out of ground effect, T_h OGE hover thrust, T_{out} actual thrust realized in IGE, T_{in} nominal thrust predicted by the OGE, h height of the rotor disk plane AGL [m], R rotor radius, V forward horizontal [m/s], V_h induced velocity in OGE hover [m/s], C_L blade-element lift coefficient.

Model	**Multi-rotor**	**Forward Flight**	**Ref.**
$\frac{T_{IGE}}{T_{OGE}} = \frac{1}{1-c(h/D)^2}\frac{1}{1+(H/V)^2}$	No	Yes	Cheeseman [19]
$\frac{T_{IGE}}{T_{OGE}} = \left(0.9926 + 0.03794\frac{(x/2r)^2}{C_L}\right)^{2/3}$	No	No	Hayden [24]
$\frac{T_{output}}{T_{input}} = \frac{1}{1-\rho\left(\frac{h}{r}\right)^2}$	Yes	No	Danjun [25]
$\frac{T_{IGE}}{T_h} = \frac{1-\frac{3r}{35}}{1+\frac{3}{50}\left(\frac{V}{V_h}\right)^3}$	Yes	Yes	Kan [26]

The variables $\{e_\phi, e_\theta, e_\psi\}$ are the attitude errors and $\{p, q, r\}$ are the angular rates. The RBFNN uses a set of Gaussian basis functions that approximate the non-linear mapping from the error space to the controller gains. By training or adaptively tuning these basis functions, the network can:

- Compensation for unmodeled dynamics, such as strong coupling between roll and pitch.
- Respond to changes in operating conditions, including different flight regimes or disturbances.
- Prevent excessive control actions by appropriately distributing or scaling gains.

With these gains based on the neural network, the torque commands for the roll, pitch, and yaw channels become Eq. (6):

$$\begin{aligned}\tau_\phi &= \left[\left(k_{p_\phi}e_\phi + k_{d_\phi}\left(\dot{\phi}_r - p\right)\right) - \hat{d_\phi}\right] b_{0,\phi} \\ \tau_\theta &= \left[\left(k_{p_\theta}e_\theta + k_{d_\phi}\left(\dot{\theta}_r - q\right)\right) - \hat{d_\theta}\right] b_{0,\theta} \\ \tau_\psi &= \left[\left(k_{p_\psi}e_\psi + k_{d_\psi}\left(\dot{\psi}_r - r\right)\right) - \hat{d_\psi}\right] b_{0,\psi}\end{aligned} \tag{6}$$

4 NUMERICAL VALIDATION

This section presents a performance evaluation of the proposed RBFNN-ADRC-GESO controller through four simulation scenarios. Its effectiveness and robustness are evaluated by comparing it with the conventional ADRC-ESO controller under external disturbances and internal model uncertainties. The simulations assume that the quadrotor starts from an initial position and attitude of $[0, 0, 0]$, with system parameters detailed in Table 2.

Table 2: The parameter values of the quadrotor UAV.

Parameter	**Value**	**Parameter**	**Value**
m (kg)	1.5	I_x (kg/m^2)	0.16
g (m/s^2)	9.8	I_y (kg/m^2)	0.16
l (m)	0.28	I_z (kg/m^2)	0.32

- *Case 1:* The UAV simulation (Fig. 2) evaluates multiaxis control performance over 60 seconds through four distinct flight phases: hovering, climbing with roll, altitude hold with pitch, and descending with yaw. Each phase introduces dynamic changes in altitude and attitude to test the system's response. Reference signals for altitude and Euler angles are adjusted per phase. The results demonstrate that the ADRC controller, enhanced with a CFD-based ground effect model, ensures accurate tracking across all axes, exhibits fast convergence, minimal overshoot, and robust handling of transient disturbances and model uncertainties near the ground level.
- *Case 2:* In this test (see Fig. 3(a)), a UAV follows a smooth ascending trajectory with lateral movement while subjected to white noise disturbances $x_d(t) = 6sin(t/3)$,$y_d(t) = -6sin(t/3)\,cos(t/3)$ and $z_d(t) = 0.8t$. The RBF-ADRC controller shows strong disturbance rejection and precise trajectory tracking, outperforming the standard ADRC controller, which shows oscillations and deviations. The results confirm the effectiveness of integrating RBFNN-based adaptive gain tuning into the ADRC framework.
- *Case 3:* To assess the robustness of the proposed control framework under model uncertainties (see Fig. 3(b)), a UAV follows a 3D helical trajectory while its physical parameters are reduced by 30% to simulate model mismatch. The RBFNN-ADRC controller, using a GESO-enhanced altitude loop and adaptive gain tuning in the attitude channel, maintains accurate tracking despite uncertainties. In contrast, the standard ADRC shows increased error. The results confirm that the RBFNN-ADRC structure enhances robustness and performance under dynamic and uncertain conditions. The reference path is defined as $x_d(t) = Rcos(\omega t)$, $y_d(t) = Rsin(\omega t)$ and $z_d(t) = z_0 + ct$, where R = 20 m, ω = 0.2 rad/s, c = 1 and z_0 = 10 m.
- *Case 4:* The target trajectory begins with a vertical ascent for the first 5 seconds (see Fig. 3(c)), followed by a circular path in the horizontal plane at a constant altitude. System robustness is tested by applying a Gaussian-shaped wind disturbance along the z-axis, Eq 7.

$$a_{wind,y}(t) = A \cdot exp\left(-\frac{1}{2}\left(\frac{t - t_0}{\sigma_t}\right)^2\right) \tag{7}$$

4.1 RBFNN-ADRC Control of UAVs in Ground Effect: Discussion

The proposed RBFNN-ADRC control strategy significantly improves the performance of the UAV in ground-effect scenarios compared to traditional ADRC. Evaluated across three flight scenarios

http://www.imavs.org/

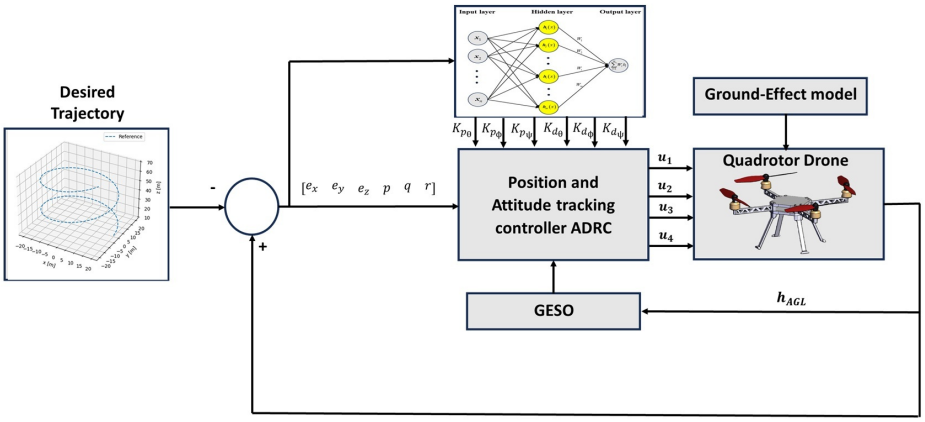


Figure 1: The diagram illustrates the control framework combining a GESO with an ADRC augmented by an RBFNN. The system tracks a predefined 3D trajectory by computing tracking errors $\begin{bmatrix} e_x & e_y & e_z & p & q & r \end{bmatrix}$, which the RBFNN processes to tune attitude control adaptively gains $\left(k_{p_\theta}, k_{d_\theta}, k_{p_\phi}, k_{d_\phi}, k_{p_\psi}, k_{d_\psi}\right)$. These gains are fed into the ADRC to generate control inputs (u_1, u_2, u_3, u_4) for the quadrotor drone. The GESO block estimates altitude states, measured altitude above ground (AGL) from the downward range sensor h_{AGL} and compensates for ground effect disturbances in real time, enhancing the controller's robustness near surfaces. The closed-loop feedback integrates drone dynamics, ground effect modeling, and adaptive gain scheduling to ensure stable and accurate flight performance.

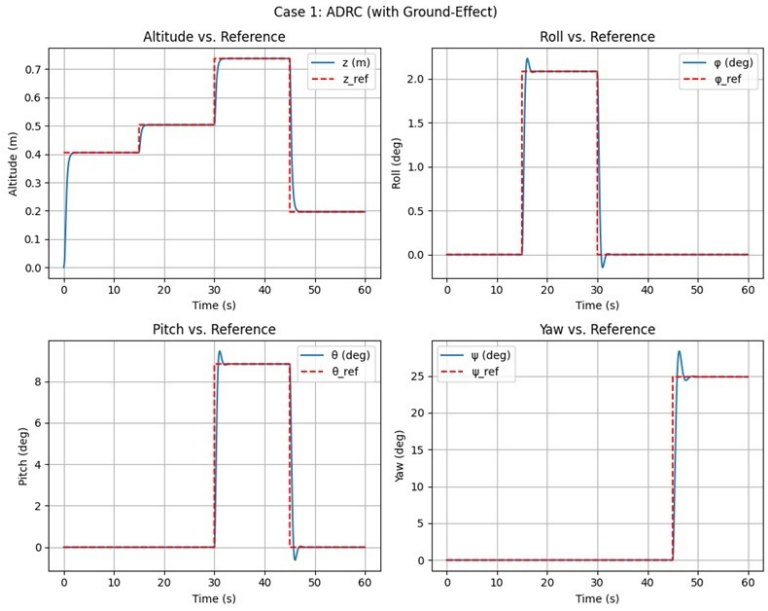


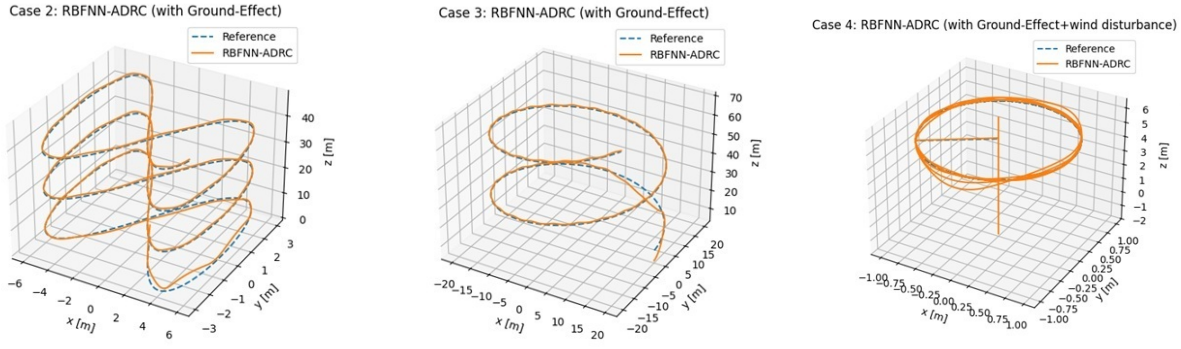
Figure 2: In this scenario, the UAV follows a phase-based reference trajectory over a 60-second simulation, divided into four distinct operational phases: hovering, ascending, pitching/rolling maneuvers, and descending with yaw rotation.

using error metrics (ISE, ITSE, IAE, ITAE), RBFNN-ADRC consistently shows superior tracking accuracy and robustness, particularly under dynamic disturbances. For example, in Scenario 1, the ISE is compared to ADRC, and in Scenario 3, it achieves a much lower ITAE despite external perturbations. Further analysis of ground-effect models within the RBFNN-ADRC framework reveals that the Dnajun model yields the best control accuracy. These findings highlight the benefit of integrating adaptive neural controllers with accurate physical models to enhance UAV stability in low-altitude environments.

5 CONCLUSION

This work introduces a robust control strategy that combines ADRC with GESO and RBFNN tuning to effectively counteract ground-effect disturbances in UAVs. A key innovation is the integration of a CFD-based empirical model for ground effect, allow-

http://www.imavs.org/



(a) (b) (c)

Figure 3: Trajectory Tracking and Disturbance Rejection Using RBFNN-ADRC with Ground Effect: (a) Case 2 $x_d\,(t) = 6sin\,(t/3)$,$y_d\,(t) = -6sin\,(t/3)\,cos\,(t/3)$ and $z_d\,(t) = 0.8t$, incorporating lateral motion, (b) Case 3- controller configurations are preserved, while the altitude loop is enhanced using a Generalized Extended State Observer (GESO) informed by a CFD-based, data-driven ground-effect model, (c) Case 4- controller configurations are preserved, with the altitude control loop enhanced via a Generalized Extended State Observer (GESO), which incorporates a CFD-informed, data-driven ground-effect model to improve compensation near the ground. In addition, a lateral wind disturbance is introduced along the y axis to simulate real-world aerodynamic perturbations that affect the stability of the quadrotor.

Table 3: Evaluation of the Control Performance of the Proposed ADRC Strategy with ESO and RBFNN-Augmented GESO for Mitigating Ground Effect Disturbances.

Controller	**ISE**	**ITSE**	**IAE**	**ITAE**
$PID_{Scenario\ 1}$	xx	xx	xx	xx
$PID_{Scenario\ 2}$	xx	xx	xx	xx
$PID_{Scenario\ 3}$	xx	xx	xx	xx
$ADRC_{Scenario\ 1}$	3.26	38.25	5.23	153.25
$ADRC_{Scenario\ 2}$	2.42	35.75	5.02	167.84
$ADRC_{Scenario\ 3}$	2.57	37.24	5.84	176
$RBFNN-ADRC_{Scenario\ 1}$	1.64	14.5	1.4	80.8
$RBFNN-ADRC_{Scenario\ 2}$	1.92	23.1	2.82	70
$RBFNN-ADRC_{Scenario\ 3}$	1.2	25.6	2.93	72.5
$RBFNN-ADRC_{Scenario\ 4}$	1.34	34.6	2.83	82.5

Table 4: Performance index Ground-Effect models for RBFNN-ADRC model proposed.

Models	**ISE**	**ITSE**	**IAE**	**ITAE**
Dnajun	1.2	29.73	2.56	71.90
Hayden	1.53	28.65	2.42	71.50
Cheeseman	1.58	21.06	2.38	74.43
Kan	1.42	24.06	2.86	85.56

ing accurate lift compensation during low-altitude flight. Embedded within the control loop, this model improves precision by capturing real aerodynamic interactions. Simulations in ROS demonstrate that the proposed RBFNN-ADRC outperforms standard ADRC across all error metrics, with ITAE in one scenario nearly halved. The Dnajun model was found to offer the best balance between accuracy and stability. Overall, the approach enables scalable, real-time control for UAV missions near the ground, such as terrain following and autonomous landing.

ACKNOWLEDGEMENTS

The first author is thankful to the National Council for Technological and Scientific Research (CONACYT) No. CVU: 661331.

REFERENCES

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

[2] Suttinee Sawadsitang, Dusit Niyato, Puay-Siew Tan, and Ping Wang. Joint ground and aerial package delivery services: A stochastic optimization approach. *IEEE Transactions on Intelligent Transportation Systems*, 20(6):2241–2254, 2019.

[3] Songchao Zhang, Xinyu Xue, Chen Chen, Zhu Sun, and Tao Sun. Development of a low-cost quadrotor uav based on adrc for agricultural remote sensing. *International Journal of Agricultural and Biological Engineering*, 12(4):82–87, 2019.

http://www.imavs.org/

http://www.imavs.org/

[4] Ranganathan Rani Hemamalini, Rajasekaran Vinodhini, Balusamy Shanthini, Pachaivannan Partheeban, Mani Charumathy, and Karunakaran Cornelius. Air quality monitoring and forecasting using smart drones and recurrent neural network for sustainable development in chennai city. *Sustainable Cities and Society*, 85:104077, 2022.

[5] Carlos Osorio Quero and Jose Martinez-Carranza. Unmanned aerial systems in search and rescue: A global perspective on current challenges and future applications. *International Journal of Disaster Risk Reduction*, 118:105199, 2025.

[6] C. Paz, E. Suárez, C. Gil, and C. Baker. Cfd analysis of the aerodynamic effects on the stability of the flight of a quadcopter uav in the proximity of walls and ground. *Journal of Wind Engineering and Industrial Aerodynamics*, 206:104378, 2020.

[7] Sam Dougherty, Nay Lin Oo, and Dan Zhao. Effects of propeller separation and onset flow condition on the performance of quadcopter propellers. *Aerospace Science and Technology*, 145:108837, 2024.

[8] Mario Ramírez-Neria, Alberto Luviano-Juárez, Jaime González-Sierra, Rodrigo Ramírez-Juárez, Joaquín Aguerrebere, and Eduardo G. Hernandez-Martinez. Active disturbance rejection control for the trajectory tracking of a quadrotor. *Actuators*, 13(9), 2024.

[9] Sofiane Khadraoui, Raouf Fareh, Mohammed Baziyad, Mahmoud Bakr Elbeltagy, and Maamar Bettayeb. A comprehensive review and applications of active disturbance rejection control for unmanned aerial vehicles. *IEEE Access*, 12:185851–185868, 2024.

[10] Chin Gian Hooi, Francis D. Lagor, and Derek A. Paley. Flow sensing, estimation and control for rotorcraft in ground effect. In *2015 IEEE Aerospace Conference*, pages 1–8, 2015.

[11] Xiang He, Gordon Kou, Marc Calaf, and Kam K. Leang. In-ground-effect modeling and nonlinear-disturbance observer for multirotor unmanned aerial vehicle control. *Journal of Dynamic Systems, Measurement, and Control*, 141(7):071013, 05 2019.

[12] H. Nobahari and A.R. Sharifi. Continuous ant colony filter applied to online estimation and compensation of ground effect in automatic landing of quadrotor. *Engineering Applications of Artificial Intelligence*, 32:100–111, 2014.

[13] Iain Fairley, Benjamin J. Williamson, Jason McIlvenny, Nicholas King, Ian Masters, Matthew Lewis, Simon Neill, David Glasby, Daniel Coles, Ben Powell, Keith Naylor, Max Robinson, and Dominic E. Reeve. Drone-based large-scale particle image velocimetry applied to tidal stream energy resource assessment. *Renewable Energy*, 196:839–855, 2022.

[14] Charbel Hage, Tonino Sophy, and El Hassane Aglzim. A comprehensive study on the aerodynamic influence of stationary and moving obstacles on an isolated phantom dji 3 uav propeller. *The Journal of Engineering*, 2024(4):e12374, 2024.

[15] Jielong Cai and Sidaard Gunasekaran. Propeller ceiling effect in forward flight. *Journal of Aircraft*, 61(4):1230–1246, 2024.

[16] Xiang He and Kam K. Leang. Quasi-steady in-ground-effect model for single and multirotor aerial vehicles. *AIAA Journal*, 58(12):5318–5331, 2020.

[17] Ambar Garofano-Soldado, Antonio Gonzalez-Morgado, Guillermo Heredia, and Anibal Ollero. Assessment and modeling of the aerodynamic ground effect of a fully-actuated hexarotor with tilted propellers. *IEEE Robotics and Automation Letters*, 9(2):1907–1914, 2024.

[18] Ke Yu and David A Peters. Nonlinear three-dimensional state-space modeling of ground effect with a dynamic flow field. *Journal of the American Helicopter Society*, 50(3):259–268, 2005.

[19] Chin Gian Hooi, Francis D. Lagor, and Derek A. Paley. Flow sensing, estimation and control for rotorcraft in ground effect. In *2015 IEEE Aerospace Conference*, pages 1–8, 2015.

[20] P-M Basset, O Heuze, JVR Prasad, and M Hamers. Finite state rotor induced flow model for interferences and ground effect. In *AHS-International Forum 57*, 2001.

[21] 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.

[22] Zhenlong WU, Tianyu ZHANG, Huijun TAN, Haoyu ZHOU, Wei CHEN, and Maixiang XIE. Hovering rotor aerodynamics in extreme ground effect. *Chinese Journal of Aeronautics*, 37(7):204–219, 2024.

[23] Jielong Cai, Sidaard Gunasekaran, and Michael Ol. Effect of partial ground and partial ceiling on propeller performance. *Journal of Aircraft*, 60(3):648–661, 2023.

[24] Lokesh Silwal, Mahendra Bhagwat, and Vrishank Raghav. Aerodynamic interactions of counter-rotating coaxial rotors hovering in ground effect. *Journal of Aircraft*, 59(6):1416–1425, 2022.

[25] Li Danjun, Zhou Yan, Shi Zongying, and Lu Geng. Autonomous landing of quadrotor based on ground effect modelling. In *2015 34th Chinese Control Conference (CCC)*, pages 5647–5652, 2015.

[26] Xinyue Kan, Justin Thomas, Hanzhe Teng, Herbert G. Tanner, Vijay Kumar, and Konstantinos Karydis. Analysis of ground effect for small-scale uavs in forward flight. *IEEE Robotics and Automation Letters*, 4(4):3860–3867, 2019.

[27] Nay Lin Oo, Dan Zhao, Mathieu Sellier, and Xiran Liu. Experimental investigation on turbulence effects on unsteady aerodynamics performances of two horizontally placed small-size uav rotors. *Aerospace Science and Technology*, 141:108535, 2023.

[28] Francisco M. F. R. Gonçalves, Ryan M. Bena, and Néstor O. Pérez-Arancibia. Closed-loop stability of a lyapunov-based switching attitude controller for energy-efficient torque-input-selection during flight. In *2024 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1941–1947, 2024.

[29] Liguo Zhang, Ying Lu, Yangzhou Chen, and Nikos E. Mastorakis. Robust uniformly ultimate boundedness control for uncertain switched linear systems. *Computers Mathematics with Applications*, 56(7):1709–1714, 2008.

http://www.imavs.org/

Appendix A: The Quadrotor Model under the Ground Effect

This study investigates the aerodynamic behavior of a quadrotor drone that flows into and out of the ground effect (IGE / OGE) using SolidWorks flow simulation. With equalized thrust at a fixed rotor speed of 8000 rpm (through pitch control), the simulation shows a thrust peak at a height-to-diameter ratio (h/D) of 0.5. Various aerodynamic models, particularly Danjun and Kan, are evaluated for their suitability to model multirotor IGE dynamics [27], although they often oversimplify near-ground airflow. The simulated drone features 5-inch propellers and an L/R ratio of 2.8, with a computational domain carefully dimensioned to minimize boundary effects (see Fig. 4). Simulations solve the RANS equations using the SST turbulence model, combining k–ω and k–ϵ methods based on proximity to the ground, with numerical settings optimized for accuracy and stability (see Fig. 5).

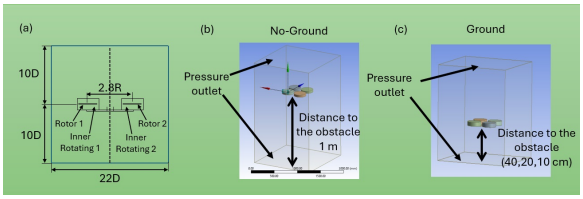


Figure 4: Simulation settings used in this study: Schematic representation, (a) quadcopter setup, (b) no-ground condition, and (c) ground condition

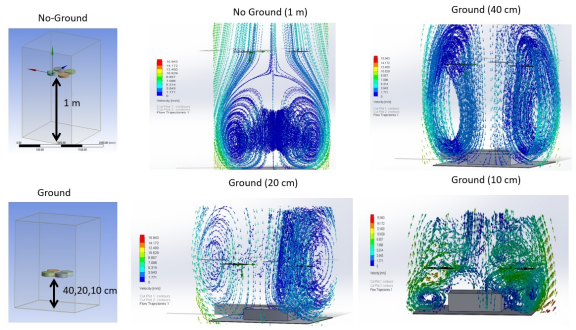


Figure 5: Simulation on Solidworks flow simulation settings study case no-ground (1 m) and ground proximity scenarios (40 cm, 20 cm, 10 cm) scenarios, operating at high rotational speeds of 8000 rpm.

Appendix B: Ground Effect Modeling using CFD-adjusted Empirical Models

An empirical model based on CFD simulation data was developed to account for the ground effect during low-altitude drone flight. The model introduces a correction factor, $M(z)$, which varies with the height of the rotor: it is interpolated for altitudes below 1.0 m and set to 1 above that threshold, where the ground effect is negligible. This approach improves simulation accuracy and supports the design of more reliable control algorithms for near-ground operations.

$$M(z) = \begin{cases} 1 & z \geq 1m \\ min\left(\frac{1}{1-\left(\frac{r}{4z}\right)^2}, 1.3 - 0.5\,(z-0.4)\right) & 0.4 \leq z < 1m \\ min\left(\frac{1}{1-\left(\frac{r}{4z}\right)^2}, 1.6 - 1.5\,(z-0.2)\right) & 0.2 \leq z < 0.4m \\ min\left(\frac{1}{1-\left(\frac{r}{4z}\right)^2}, 2.2 - 6\,(z-0.1)\right) & 0.1 \leq z < 0.2m \\ 2.2 & z < 0.1m \end{cases} \tag{8}$$

- Cheeseman model with CFD correction: $\hat{GE}_{CFD} = \hat{GE}_{Cheeseman} xM(z)$
- Hayden model with CFD correction:

$$\hat{GE}_{Hayden} = \begin{cases} \left(0.9926 + 0.03794 \frac{(z/2r)^2}{C_L}\right)^{2/3} & 0.1 \leq z < 1m \\ 2.2 & z < 0.1m \\ 1 & z \geq 1m \end{cases} \tag{9}$$

  where r is the rotor radius and C_L is the lift coefficient.
- Kan model with CFD correction: $\hat{GE}_{kan} = \left(1 - \frac{3r}{35}\right) / \left(1 + \frac{3}{50}\left(\frac{V}{V_h}\right)^3\right) xM(z)$ where V is the actual climb velocity and V_h is the hovering-induced velocity.
- Dnajun model with CFD correction: $\hat{GE}_{CFD} = \hat{GE}_{Dnajun} xM(z)$

Appendix C: Stability Analysis via Lyapunov Theory

A Lyapunov-based stability analysis is performed for the altitude control subsystem [28] to evaluate the robustness of the proposed RBFNN-ADRC-GESO control strategy against wind-induced disturbances and near-ground aerodynamic effects. Let the altitude tracking error be defined as $e_z = z(t) - z_r(t)$ and $e_{\dot{z}} = \dot{z}(t) - \dot{z}_r(t)$,where $\dot{z}(t)$ is the desired altitude trajectory and $\dot{z}_r(t)$ it is a time derivative. A candidate Lyapunov function is proposed as Eq. (10):

$$V(t) = \frac{1}{2} e_z^2(t) + \frac{1}{2} e_{\dot{z}}^2(t) \tag{10}$$

This positive definite function measures the total instantaneous tracking energy in the altitude channel. Taking its time derivative yields the following Eq. (11).

$$\dot{V}(t) = e_z(t) \cdot \dot{e}_z(t) + e_{\dot{z}}(t) \cdot \dot{e}_{\dot{z}}(t) \tag{11}$$

Substituting the tracking error dynamics governed by closed-loop altitude dynamics, and incorporating the observer-based disturbance estimation $\hat{f}_{ext}$, we obtain Eq. (12).

$$\dot{e}_z = e_{\dot{z}}, \quad \dot{e}_{\dot{z}} = \frac{1}{m} T_z - g - \ddot{z}_r(t) + \Delta(z, \dot{z}, t), \tag{12}$$

where $\Delta(.)$ includes the lumped uncertainty from ground-effect correction $M_{GE}(z)$ and wind-induced perturbations. The control input T_z is designed as Eq. (13):

$$T_z = m \cdot \left(\ddot{z}_r(t) - k_p e_z - k_d e_{\dot{z}} + \hat{f}_{ext}\right) \tag{13}$$

Substituting Eq. (12) into Eq. (11):

$$\dot{V}(t) = -k_p e_z^2 - k_d e_{\dot{z}}^2 + e_{\dot{z}} \cdot \left(\Delta - \hat{f}_{ext}\right) \tag{14}$$

Assume a bounded disturbance estimation error $\left|\Delta - \hat{f}_{ext}\right| \leq \varepsilon$, and applying Young's inequality Eq. (15) and Eq. (16).

$$e_{\dot{z}} \cdot \left(\Delta - \hat{f}_{ext}\right) \leq \frac{1}{2} k_d e_{\dot{z}}^2 + \frac{\varepsilon^2}{2k_d} \tag{15}$$

$$\dot{V}(t) \leq -k_p e_z^2 - \frac{1}{2} k_d e_{\dot{z}}^2 + \frac{\varepsilon^2}{2k_d} \tag{16}$$

APPENDIX D: LYAPUNOV STABILITY RBFNN ADAPTATION

The attitude control gains k_{pi}, k_{di} for $i \in \{\phi, \theta, \psi\}$ are adaptively generated using a Radial Basis Function Neural Network (RBFNN) Eq. (17):

$$\begin{bmatrix} k_{pi}(t) & k_{di}(t) \end{bmatrix} = \sum_{j=1}^{N} \omega_{ij} \phi_j \left(e(t)\right) \tag{17}$$

where $\phi(\cdot)$ are Gaussian basis functions on the state error vector $e = [e_i, \omega_i]^T$, where the basis functions are normalized $\sum_j \phi_j(\cdot) \leq 1$, ω_{ij} are trained weights bounded $\omega_{ij} \in [\omega_{min}, \omega_{max}]$. Hence, the adaptive gains are positive and bounded $0 < k_{pi}^{min} \leq k_{pi}(t) \leq k_{pi}^{max}$,$0 < k_{di}^{min} \leq k_{di}(t) \leq k_{di}^{max}$. Let us define a Lyapunov function for each attitude channel $V_i(t) = \frac{1}{2} e_i^2 + \frac{1}{2} \omega_i^2$, using a similar derivation $\dot{V}_i \leq (t) - \lambda_1 \|e_i\|^2 - \lambda_2 \|\omega_i\|^2 + \delta_i(t)$, where $\lambda_1, \lambda_2 > 0$ depend on the adaptive gain bounds, and δ_i is a bounded disturbance term from the modeling error or external wind. As a result, the attitude subsystem is guaranteed to eventually be uniformly bounded (UUB) [29] under the adaptive control strategy based on the RBFNN.

http://www.imavs.org/

CU-Fundamental: A Comprehensive Simulation Framework for Autonomous MAV

Jialiang Wang[1]*, Yijun Huang[1], Yizhou Chen[2], Zongzhou Wu[1], Qigeng Duan[1], Zhiyu Zhou[3], Yanqi Zhao[1], Zhipeng Lin[1], Jiwen Xu[1], Jerry Tang[4], Xi Chen[1], Zhi Gao[3] and Ben M. Chen[1]
[1]*Department of Mechanical and Automation Engineering*, The Chinese University of Hong Kong, Hong Kong SAR
[2]*School of Computing and Data Science*, Hong Kong University, Hong Kong SAR
[3]*School of Remote Sensing and Information Engineering*, Wuhan University, China
[4]*Hong Kong Centre For Logistics Robotics*, Hong Kong University, Hong Kong SAR

http://www.imavs.org/

ABSTRACT

In recent years, the development of Unmanned Aerial Vehicles (UAVs) has seen significant advancements, driven by their successful applications in various fields. Autonomous Micro Aerial Vehicles (MAVs) are particularly gaining attention due to their agility and potential for complex tasks. However, testing autonomous algorithms on real MAVs is both costly and risky, making simulation platforms essential for safe and efficient development. This paper introduces CU-Fundamental, a comprehensive simulation framework for autonomous MAVs, which supports multiple levels of simulation, including numerical simulation, Software-in-the-Loop (SITL), and Hardware-in-the-Loop (HITL). The framework integrates essential algorithms facilitating autonomous flight, including pose estimation, map building, path planning, motion planning, and low-level control. CU-Fundamental is designed to be modular and flexible, allowing for easy integration of new algorithms and components. It employs Robot Operating System (ROS) for internal communication, a web-based Ground Control Station (GCS) for advanced user interaction, PX4 firmware for flight control, and Gazebo for realistic physical simulation. By providing a flexible and modular architecture, CU-Fundamental aims to accelerate the development of robust autonomous MAV systems while minimizing risks and costs associated with real-world testing.

1 INTRODUCTION

In recent years, the development of UAVs has witnessed remarkable progress, with their applications expanding across a wide range of fields, including search and rescue, environmental monitoring, last-mile delivery, building inspection and autonomous racing [1]. Particularly, Autonomous MAVs have garnered significant attention due to their agility and potential to perform complex tasks in confined spaces such as tunnels and mines [2]. These advancements have not only enhanced the efficiency of various operations but also opened up new possibilities for innovation in the field of aerial robotics.

However, the increasing complexity of tasks that UAVs are expected to perform necessitates the development and integration of sophisticated algorithms. These algorithms encompass a wide range of functionalities, including low-level control laws for precise maneuvering, robust pose estimation for autonomous flight, accurate perception for environmental awareness, and high-level planning algorithms for optimal path generation. Validating the effectiveness of these algorithms could be costly and sometimes rather dangerous on real UAVs [3]. Therefore, simulation platforms have emerged as a crucial solution, providing a safe and efficient means to test and refine autonomous algorithms before deployment on actual platforms.

Several simulation programs have been developed to address the needs of UAV research and development, among which AirSim and PX4 Simulink are two popular platforms that have proved to be effective in various scenarios. Developed by Microsoft, AirSim is a high-fidelity simulation platform that provides realistic rendering of environments and sensor data [4]. On the other hand, PX4 Simulink focuses on the development and testing of low-level control algorithms for UAVs [5]. It provides a comprehensive environment for designing, simulating, and tuning control systems, leveraging the powerful capabilities of Simulink for model-based design and verification.

While these platforms have been instrumental in advancing UAV research, they also have limitations that need to be addressed. AirSim, for instance, excels in providing realistic sensor data, making it suitable for testing high-level perception algorithms. However, the platform is not easily extensible for secondary development, partially due to the heavy Unreal Engine it is built upon. It also has limited support for Linux, which restricts its usability makes integration with existing algorithms very challenging. Similarly, PX4 Simulink enables researchers to develop robust and reliable control algorithms through its high-performance numerical simulation. Yet, when it comes to solving practical problems that involve

*Email address: jialiang.wang@link.cuhk.edu.hk

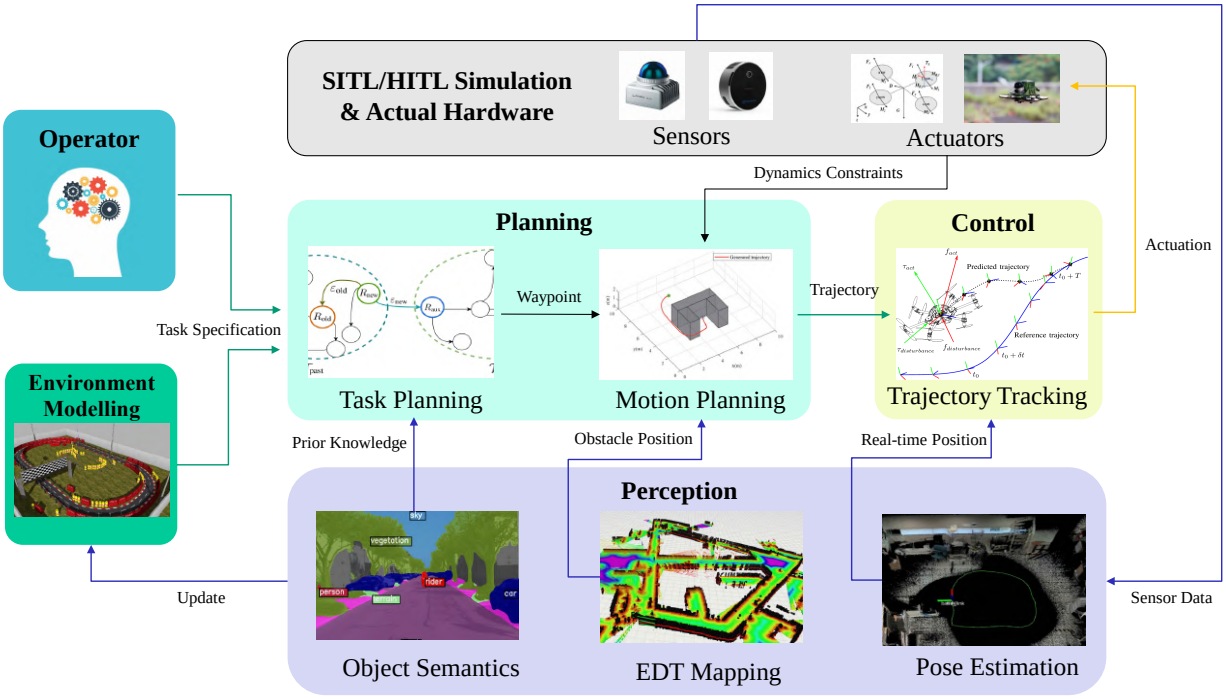


Figure 1: The architecture of CU-Fundamental. The system has built-in modules for planning, control and perception, with standard interfaces facilitating smooth communication between modules.

higher-level functionalities such as Simultaneous Localization and Mapping (SLAM) and planning, Simulink is not suitable, if not impossible, to use due to its limited capability to simulate sensor data and the complexity of integrating other algorithms into Matlab.

To test both low-level and high-level algorithms for autonomous MAVs, we introduce CU-Fundamental, a comprehensive simulation framework for autonomous MAVs. CU-Fundamental is designed to support multiple levels of simulation, including numerical simulation, SITL, and HITL, following the implementation of XTDrone [6]. It employs ROS for internal communication, a GCS for advanced user interaction, PX4 firmware for flight control, and Gazebo for realistic physical simulation. The framework has built-in support for essential algorithms like SLAM, path planning, motion planning, and control. Moreover, CU-Fundamental is designed to be modular and flexible, providing standard interfaces for integration of new algorithms and fine-tuning of algorithm parameters. This framework facilitates the fundamental workflow of developing and testing autonomous MAVs: the upper-level algorithms are first tested (in SITL mode) while taking aerodynamics (in numeric simulation) into account, then the same algorithms are deployed to the embedded flight control board and onboard computer for HITL testing. If all tests are successful, the algorithms can be deployed to real MAVs for field-testing.

2 System Architecture

An overview of the architecture of CU-Fundamental is shown in Figure 1.

2.1 *Dynamic Modelling*

Upon finishing the design of new MAVs, the first step is to model the dynamics of the MAV. Taking advantage of Gazebo's physics engine [7], CU-Fundamental provides a dynamic model of quadrotor MAVs as shown in Figure 2.

The forces and moments on an MAV comprise the thrust force $f_i \in \mathbb{R}$ and corresponding moment $M_i \in \mathbb{R}^3$ generated by i-th rotor, the aerodynamic drag force $D \in \mathbb{R}^3$ and the gravitational force $G \in \mathbb{R}^3$. Let the distance from the center of mass to the center of each rotor be $d \in \mathbb{R}$, the total thrust f and moment $M = [M_x, M_y, M_z]^T$ generated in the body frame can be expressed as:

$$\begin{bmatrix} f \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -d & 0 & d \\ d & 0 & -d & 0 \\ -c_M & c_M & -c_M & c_M \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \quad (1)$$

where c_M is the moment coefficient, which is a constant determined by the geometry of the MAV. More detailed modelling of this coefficient can be found in pp. 631-639 of [8].

Applying Newton's second law, the equations of motion

http://www.imavs.org/

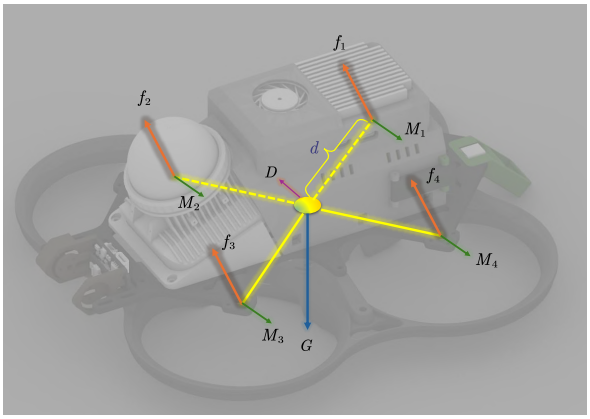


Figure 2: The dynamic model of quadrotor MAVs in CU-Fundamental.

of the quadrotor MAV can be expressed as:

$$\dot{x} = v, \tag{2}$$

$$m\dot{v} = G + D + f, \tag{3}$$

$$\dot{R} = R\hat{\Omega}, \tag{4}$$

$$J\dot{\Omega} + \Omega \times J\Omega = M, \tag{5}$$

where $R \in SO(3)$ is the rotation matrix from body frame to world frame, $\Omega \in \mathbb{R}^3$ is the angular velocity in body frame, $J \in \mathbb{R}^{3\times3}$ is the inertia matrix of the MAV, and $\hat{\Omega}$ is the skew-symmetric matrix of Ω.

2.2 *Sensor Simulation*

Sensor models are essential for the perceptual capability of autonomous MAVs. To cater for the versatile needs of various missions, CU-Fundamental provides a set of different sensors including IMU, GPS, LiDAR, and cameras.

IMU, magnetometer, barometer and GPS: Following the design of [6], the observation values of all these sensors are disturbed by an error model which consists of a random walk, a bias and a white noise. These parameters are all configurable.

Camera: CU-Fundamental supports general RGB and depth cameras provided by Gazebo plugins. The tunable parameters include camera resolution, field of view (FOV), intrinsic parameters, distortion coefficients, among others. Alternatively, users may opt to use other open-source camera simulation plugins such as the Intel Realsense simulation by [9], which essentially combines multiple RGB and depth cameras into a single plugin.

LiDAR: CU-Fundamental supports various types of LiDAR sensors, including 2D and 3D ones. 2D LiDAR is implemented as an instance of Gazebo ray sensor, which emits laser beams controlled by a certain range, FOV, resolution, number of samples and Gaussian noise. Two types of 3D LiDAR are supported: the first is mechanical LiDAR, which is implemented as a vertical array of 2D laser beams; the second is solid-state LiDAR, which is based on the working principle of Livox LiDARs [10].

2.3 *Supporting Algorithms*

Control: CU-Fundamental allows users to implement both outer-loop and inner-loop control algorithms using the functionalities of Mavros. By default, CU-Fundamental uses PX4 for attitude, position and velocity control. Attitude control is achieved using traditional PID control law, while position and velocity control are implemented as cascaded PID controllers on top of the attitude controller.

Moreover, CU-Fundamental offers a more sophisticated non-linear geometric controller [11, 12] as an in-place alternative to the default PID controller. Similarly, three flight modes, namely attitude controlled mode, position controlled mode and velocity controlled mode, are supported and can be switched between each other. Different from the default PID controllers, the non-linear geometric controller guarantees almost global tracking features on $SO(3)$ and enables more aggressive maneuvers with faster response time.

Perception: Perception is a board term that refers to the process of acquiring and interpreting sensory data to gain actionable insights about the environment. Among various perceptual algorithms, SLAM is one of the most crucial components because of its capability to provide accurate pose estimation and environmental mapping for both upper-level planning algorithms and low-level control. Depending on the sensor types, CU-Fundamental has built-in support for various visual and LiDAR SLAM algorithms, providing convenient choices for users. For visual SLAM, CU-Fundamental supports the ORB-SLAM series [13, 14, 15] and the VINS series [16, 17], which are both well-known open-source visual SLAM algorithms. For LiDAR SLAM, CU-Fundamental implements the LOAM family [18] and Fast-LIO [19].

Additionally, some specific mapping algorithms are included as a middle layer between SLAM and planning algorithms. For example, our previous work [20] provides a GPU-accelerated Euclidean distance transform algorithm to enhance environmental awareness of the obstacles, ensuring better motion planning performance. Finally, AI-powered object detection and segmentation algorithms are also supported to provide semantic understanding of the world. CU-Fundamental includes the YOLO series [21, 22] and provides corresponding interfaces.

Planning: A hierarchical planning framework based on our previous work [23] is implemented in the system, which consists of a task planner, a path planner and a motion planner. A task planner based on linear temporal logic [24] is first called to generate a sequence of waypoints based on mission specifications. The path planner then searches for a feasible path, which is subsequently smoothed by a motion planner to generate a trajectory consisting of a sequence of desired poses, velocities and accelerations. Moreover, CU-Fundamental includes popular motion planning algorithms

http://www.imavs.org/

such as Ego-planner [25].

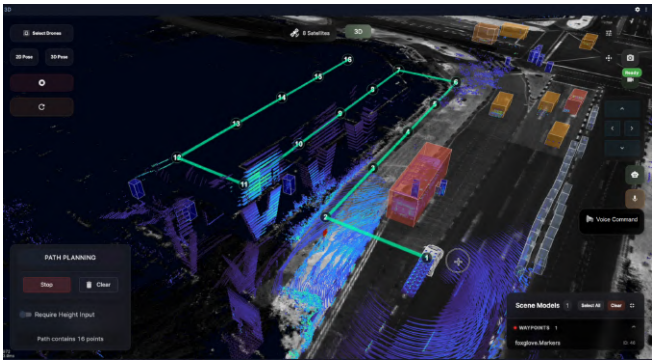


Figure 3: A Snapshot of AstroGCS visualizing the nuScenes dataset [26]. The waypoints are generated and published to onboard computer using large language models with the following prompt: "Generate a zigzag path with three rows, each row containing five waypoints".

2.4 *Interface and Communication*

The communication between different algorithms is facilitated by ROS, whose message-passing mechanism allows for efficient data exchange. In addition to the standard messages, CU-Fundamental provides a set of custom messages[1] to handle the data flow between SLAM, planning and control algorithms.

In terms of lower-level communication with the simulated flight control board, Mavlink and Mavros are used to ensure compatibility with the PX4 firmware. For example, the sensor data and MAV dynamic status simulated by Gazebo plugins are published to the flight controller through Mavlink, while other necessary information (localization results, user control commands, etc.) that are calculated in ROS are sent through Mavros. For detailed information, readers are referred to [6].

On the user side, CU-Fundamental provides a more user-friendly and modernized GCS named AstroGCS using the web-based (HTML5 and React) architecture. The GCS provides a 3D graphical interface for users to monitor the status of the MAV, visualize the environment, and interact with the system. As is illustrated in Figure 3, it also supports advanced features such as publishing planning commands and invoking AI agents to assist in decision-making. AstroGCS has access to all the data communicated in ROS through ROS Bridge (See Figure 4). Compared to Qt-based counterparts, the web-based design is easier to program and extend thanks to the rich ecosystem of web technologies. Moreover, since it is web-based, the GCS can be deployed on any device with a modern web browser and could be even accessed remotely, suggesting better cross-platform compatibility and more convenient user experience.

[1] https://github.com/CUHK-UAS-FUNDAMENTAL/usrl_msgs/

One shortcoming of AstroGCS is that it does not support showing the Gazebo simulation since Gazebo adopts a different communication and rendering strategy. Gazebo uses a server-client architecture, where the server (GzServer) is responsible for simulating the physics while the client (GzClient) is for rendering the simulation results. Since GzClient is not web-based, it cannot be directly integrated into AstroGCS. Thankfully, Gazebo provides GzBridge and GzWeb that allow users to visualize the simulation results in a web browser. CU-Fundamental modifies the GzWeb to provide advanced functionalities such as manipulating MAV models. In this way, GzWeb and AstroGCS work in accordance to provide a unified system UI in web architecture, where users could monitor the simulation results in GzWeb and manipulate drone algorithms in AstroGCS. Altogether, the architecture of the interface and communication is shown in Figure 4.

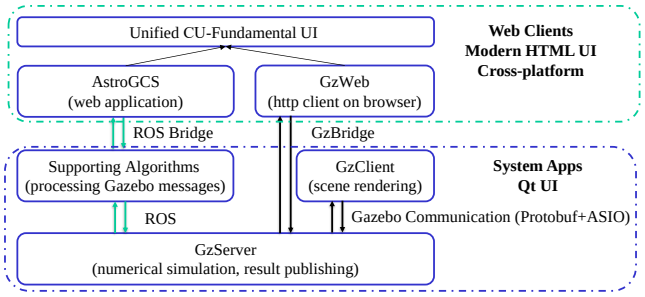


Figure 4: Communication and interface between AstroGCS, algorithms and Gazebo simulation. Components in the blue box are readily implemented in ROS and adopted by most simulation frameworks. CU-Fundamental provides additional unified web interface as marked in the green box.

3 Case Study

This study reproduces the tunnel navigation challenge in the IMAV 2025 competition, where the MAV is required to navigate through five 0.5 m × 1 m tunnels spanning a total length of 2 m. A new MAV named CU-Astro is designed for this task, which is a quadrotor MAV with a size of 216 mm × 216 mm × 95 mm. The MAV is equipped with a 3D LiDAR (Livox Mid-360) for LiDAR SLAM and detection of the tunnel walls. The overall design of CU-Astro is illustrated in the background of Figure 2.

Two necessary testing stages are required to complete the task. Firstly, the MAV should be stable enough to track a given trajectory, which is primarily achieved by tuning the control parameters. Subsequently, the high-level detection and planning algorithms are tested to automatically generate a feasible trajectory for navigation.

3.1 *Minimum-snap Trajectory Generation and Tracking*

To ensure basic stability and controllability of the MAV, a minimum-snap trajectory is generated for the drone to follow [27]. Minimum-snap trajectory is a widely used

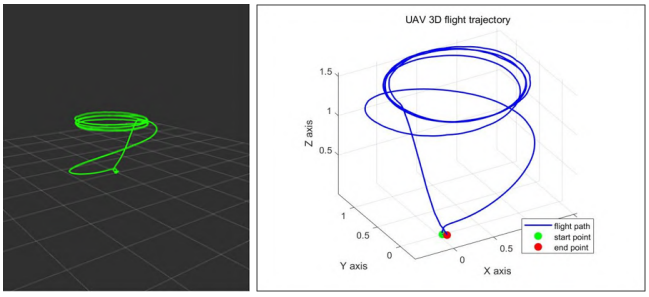
http://www.imavs.org/

method for generating smooth flight paths for MAVs and integrated in various motion planning algorithms included in CU-Fundamental. Therefore, successful tracking of minimum-snap trajectory suggests easier integration of other algorithms in the future. Specifically, the problem could be formulated as the following optimization objective:

$$\min_{\mathbf{p}(t)} \int_{t_0}^{t_f} \alpha \left\| \mathbf{p}^{(n)}(t) \right\|^2 + \beta \left\| \Psi^{(m)}(t) \right\| dt, \quad (6)$$
$$\text{s.t. } \mathbf{p}(t_0) = \mathbf{p}_0, \mathbf{p}(t_f) = \mathbf{p}_f, \Psi(t_0) = \Psi_0, \Psi(t_f) = \Psi_f$$

where $\mathbf{p}(t) \in \mathbb{R}^3$ is the position of the MAV at time t, $\Psi(t)$ is the yaw angle, $\mathbf{p}_0$ and $\mathbf{p}_f$ are the initial and final positions, Ψ_0 and Ψ_f are the initial and final yaw angles. n and m control the order of derivatives. α and β are scaling factors. For minimum-snap trajectory generation, $n = 4$ and $m = 2$ are used, which results in minimized differential thrust in the physical system, thereby reducing the energy consumption and improving the smoothness of the flight. To track the generated trajectory, a typical cascaded PID controller is used.

In the simulation stage, a minimum-snap trajectory is generated to pass through four waypoints located at the corners of a square with a size of 1 m × 1 m. The square is placed on a horizontal plane at a height of 1.5 m.



(a) Simulation (b) Real flight

Figure 5: Minimum-snap trajectory generated and tracked in simulation and real flight.

After the workflow of the algorithms is successfully verified, the same algorithms are deployed to the onboard computer. The control parameters used in the simulation serve as the initial values for the real flight and are further fine-tuned using the PX4 Autotune. Figure 5 shows the minimum-snap trajectory generated and tracked in both simulation and real flight. It is noted that these two trajectories are highly similar, indicating that the algorithms are successfully transferred from simulation to real flight. The real flight data is recorded by PX4 and shown in Figure 6. While there's some delay in the response of the MAV, the final trajectory aligns well with the planned.

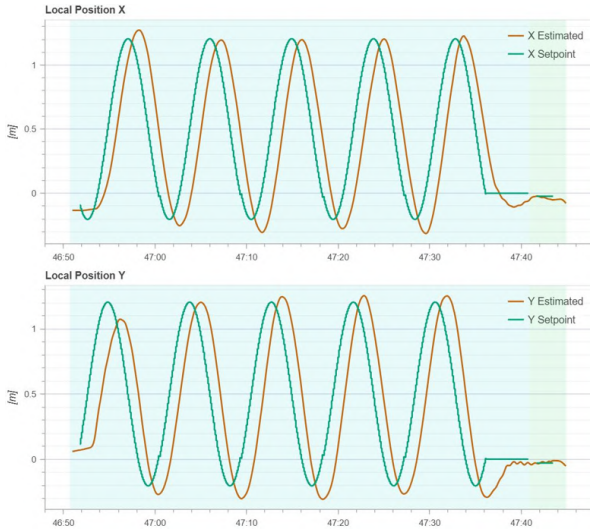


Figure 6: Real flight data record of tracking results.

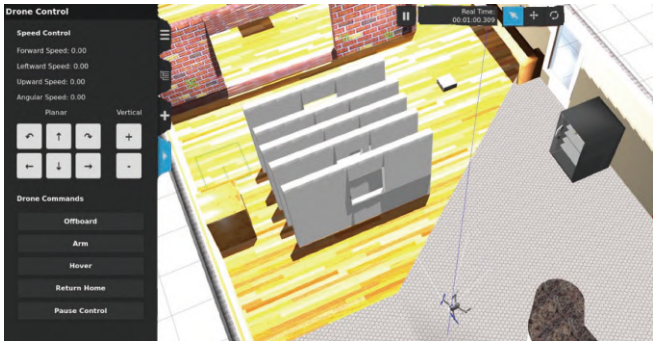


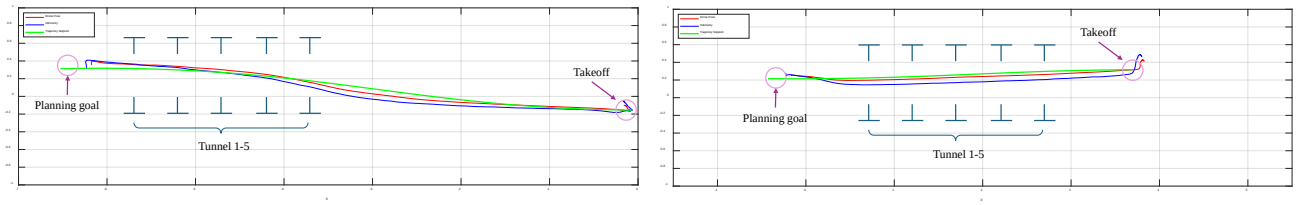
Figure 7: Gazebo simulation of the tunnel navigation challenge. The scene rendering is captured from GzWeb. The left side menu also shows a part of the GzWeb interface, which allows users to manipulate the MAV directly and visualize the simulation results.

3.2 *Tunnel Detection and Navigation*

After the MAV's mobility is verified, the next is to integrate high-level algorithms including object detection, SLAM and motion planning algorithms are tested to find feasible solutions. In the simulation stage, a Gazebo environment as shown in Figure 7 is first created to simulate the tunnel environment. Then the MAV is manually guided using proper external localization and motion planner. As CU-Fundamental has built-in algorithms and interfaces for these functionalities, the development process is significantly accelerated. Preliminary experiments suggest that a combination of Fast-LIO2, Ego-planner and PID controller can achieve successful navigation provided that correct waypoints are given.

To automatically generate waypoints, a simple object detection method based on 3D point cloud is implemented. The algorithm first filters the point cloud to remove noisy points,

http://www.imavs.org/



(a) Simulation (b) Real flight

Figure 8: Visualization of the trajectory in simulation and real flight. The drone pose (red), odometry (blue) and trajectory setpoints (green) are shown in the figure. The Z-axis is omitted in the plot since the tunnel is high enough to tolerate relatively large altitude errors.

then applies a DBSCAN clustering algorithm to detect and segment the tunnel walls. The center of the tunnel is then estimated as the average of the points in each cluster. After the MAV is guided to the center of the first tunnel wall, the location of the furthermost center is used as the planning goal.

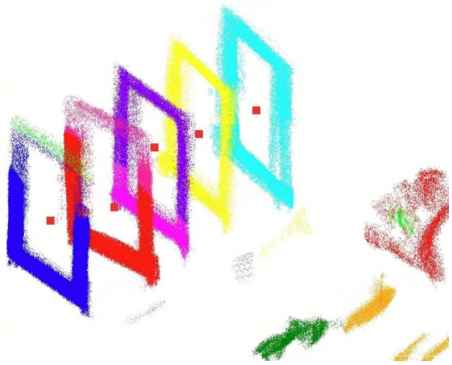


Figure 9: DBSCAN clustering of tunnel walls. The center is marked as red points.

The aforementioned algorithms are first tested in simulation and then deployed for real flight. Since the detection algorithm is based on the relatively easy-to-simulate LiDAR sensor (only geometry of the pointcloud is required), there is little simulation-to-real gap. The final flight trajectory is illustrated in Figure 10.

To qualitatively evaluates the function of CU-Fundamental, Figure 8 compares the odometry, actual drone pose and trajectory setpoints in both simulation and real flight. The statistics of the corresponding errors are reported in Table 1. Odometry error is defined as the Euclidean distance between the drone pose and the odometry while tracking error is that between the drone pose and the trajectory setpoints.

As is indicated in the figure and table, the error pattern in real flight is similar to that in simulation, suggesting the practical usability of CU-Fundamental as a simulation framework for real-world applications. Both odometry and tracking errors are statistically below 10 cm, which is acceptable for the task since the drone size is 21.6 cm compared to the tunnel size of 50 cm. The maximum tracking error is recorded at the end of the flight in both cases, which is likely due to the accumulated error in the odometry. To overcome this issue, it is advised that the MAV should adopt some advanced algorithms to readjust its location with respect to the final tunnel.



Figure 10: Composite image of the drone trajectory.

Table 1: Odometry error and tracking error (Unit: m)

Error type	Odometry error		Tracking error	
	simulation	real	simulation	real
Min	0.0217	0.0447	0.0037	0.0022
RMSE	0.0962	0.0855	0.0316	0.0655
Max	0.2577	0.1449	0.0575	0.1647

4 Conclusion

This paper presents CU-Fundamental, a comprehensive simulation framework for autonomous MAVs. The framework supports multiple levels of simulation, including numerical simulation, SITL, HITL, and integrates essential algorithms for autonomous flight. CU-Fundamental is based on modular design, implementing standard interfaces for various SLAM, planning and control algorithms. It also innovates the user interface by providing web-based AstroGCS

http://www.imavs.org/

and GzWeb, which allows for advanced user interaction and easier secondary development. The framework is tested in a case study of tunnel navigation, demonstrating its effectiveness in simulating and testing autonomous MAV algorithms. The future work will focus on further enhancing the modularity and extensibility of CU-Fundamental, as well as improving simulation realism to better reflect real-world scenarios.

ACKNOWLEDGEMENTS

This work was supported in part by the InnoHK of the Government of the Hong Kong Special Administrative Region via the Hong Kong Centre for Logistics Robotics and in part by the Research Grants Council of Hong Kong SAR (Grant Nos: 14217922, 14209623 and 14209424).

REFERENCES

[1] Mostafa Hassanalian and Abdessattar Abdelkefi. Classifications, applications, and design challenges of drones: A review. *Progress in Aerospace sciences*, 91:99–131, 2017.

[2] Julius A Marshall, Wei Sun, and Andrea L'Afflitto. A survey of guidance, navigation, and control systems for autonomous multi-rotor small unmanned aerial systems. *Annual Reviews in control*, 52:390–427, 2021.

[3] Adriano Bittar, Helosman V Figuereido, Poliana Avelar Guimaraes, and Alessandro Correa Mendes. Guidance software-in-the-loop simulation using x-plane and simulink for uavs. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 993–1002. IEEE, 2014.

[4] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics: Results of the 11th international conference*, pages 621–635. Springer, 2017.

[5] 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. IEEE, 2015.

[6] Kun Xiao, Shaochang Tan, Guohui Wang, Xueyan An, Xiang Wang, and Xiangke Wang. Xtdrone: A customizable multi-rotor uavs simulation platform. In *2020 4th International Conference on Robotics and Automation Sciences (ICRAS)*, pages 55–61. IEEE, 2020.

[7] Russell Smith et al. Open dynamics engine. 2005.

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

[9] PAL Robotics. Intel realsense gazebo ros plugin, 2019. Available at `https://github.com/pal-robotics/realsense_gazebo_plugin`. Last accessed: 2025-07-08.

[10] Francesco Vultaggio, F d'Apolito, C Sulzbachner, and P Fanta-Jende. Simulation of low-cost mems-lidar and analysis of its effect on the performances of state-of-the-art slams. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 48:539–545, 2023.

[11] Taeyoung Lee, Melvin Leok, and N Harris McClamroch. Control of complex maneuvers for a quadrotor uav using geometric methods on se (3). *arXiv preprint arXiv:1003.2005*, 2010.

[12] Taeyoung Lee, Melvin Leok, and N Harris McClamroch. Geometric tracking control of a quadrotor uav on se (3). In *49th IEEE conference on decision and control (CDC)*, pages 5420–5425. IEEE, 2010.

[13] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: A versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[14] 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.

[15] 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.

[16] 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.

[17] Tong Qin, Jie Pan, Shaozu Cao, and Shaojie Shen. A general optimization-based framework for local odometry estimation with multiple sensors, 2019.

[18] Ji Zhang, Sanjiv Singh, et al. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and systems*, volume 2, pages 1–9. Berkeley, CA, 2014.

[19] Wei Xu and Fu Zhang. Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter. *IEEE Robotics and Automation Letters*, 6(2):3317–3324, 2021.

http://www.imavs.org/

[20] Yizhou Chen, Shupeng Lai, Jinqiang Cui, Biao Wang, and Ben M Chen. Gpu-accelerated incremental euclidean distance transform for online motion planning of mobile robots. *IEEE Robotics and Automation Letters*, 7(3):6894–6901, 2022.

[21] Marko Bjelonic. YOLO ROS: Real-time object detection for ROS, 2016–2018. Available at `https://github.com/leggedrobotics/darknet_ros`. Last accessed: 2025-07-08.

[22] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[23] Lele Xi, Xinyi Wang, Lei Jiao, Shupeng Lai, Zhihong Peng, and Ben M Chen. Gto-mpc-based target chasing using a quadrotor in cluttered environments. *IEEE Transactions on Industrial Electronics*, 69(6):6026–6035, 2021.

[24] Yizhou Chen, Ruoyu Wang, Xinyi Wang, and Ben M Chen. Sampling-based path planning under temporal logic constraints with real-time adaptation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3700–3706. IEEE, 2023.

[25] 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, 2020.

[26] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020.

[27] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525, 2011.

http://www.imavs.org/

A Comparative Study on Thrust Map Estimation for Multirotor Aerial Vehicles

Francisco J. Anguita [1], Rafael Perez-Segui [1], Carmen DR.Pita-Romero [1], Miguel Fernandez-Cortizas [1,2], Javier Melero-Deza [1], Pascual Campoy [1] *

Abstract

Precise drone flight requires stable low-level control. Direct control over the drone's collective thrust is unfeasible in most cases. A Thrust Map (TM) can be calibrated as a means of accurately actuating the motors. In this work, different approximations to a TM are evaluated to develop a general methodology for offline calibration in two steps. First, an offline estimation is obtained from experimental preflight data. Second, the offline TM is refined using a Correction Factor computed from real flight data. Different TM strategies are then discussed in terms of their performance and the resources required to compute them. Extensive experimental validation of the proposed methods has been conducted, demonstrating the effectiveness of a TM calibrated from thrust stand experiments and adjusted using the proposed correction factor. A repository with code and a video explanation are attached to help replicate this work.[1] [2]

http://www.imavs.org/

1 Introduction

Quadrotors excel for their great agility and speed. These characteristics make them ideal for demanding applications, such as drone racing or ship landing in maritime harsh environments, which require precise maneuvers [1, 2]. Such agile flight requires fast and robust low-level control, particularly attitude or rate control [3], which involves angular position (attitude), angular velocity (rates) and collective thrust commands. However, closing the control loop on thrust directly requires force sensors, increasing equipment costs and reducing flight time [4].

To avoid the drawbacks of onboard thrust sensing, quadrotors typically rely on electric propulsion systems—specifically, Brushless Direct Current (BLDC) motors—controlled through Electronic Speed Controllers (ESCs) [5]. Fuel-based propulsion systems are largely unsuitable for small quadrotors due to their weight and decreasing environmental acceptance. Flight controllers convert attitude and rate commands into throttle signals, which are normalized control inputs sent to the ESCs to actuate the rotors. Table 1 lists commonly used Flight Controller Units (FCUs) and the structure of their throttle commands.

FCU	Signal Technique	Throttle Range
Betaflight	PWM	1000-2000μs
	Oneshot	125-250μs
Ardupilot	PWM	1000-2000μs
	Oneshot	125-250μs
Pixhawk/PX4	PWM	1000-2000μs
	Oneshot	125-250μs
Crazyflie	Custom scale	10001-60000
Parrot	App slider	UI-based normalized %
Skydio	App slider	UI-based normalized %

Table 1: Throttle command encoding methods and ranges for popular flight controllers units. The *Signal Technique* indicates how throttle commands are transmitted to the ESCs, while the *Throttle Range* specifies the corresponding signal values or scales used to represent throttle percentages.

A critical component of this control chain is the relationship between the throttle input and the actual thrust produced by the rotors. The force F produced by the rotor depends on its characteristics and the angular velocity of the motor ω. This relationship can be modeled as shown in Equation 1 [6], using a thrust coefficient k_f.

$$F = k_f \cdot \omega^2 \tag{1}$$

Directly measuring this force, or estimating it from the rotor's angular speed using Equation 1, would require additional sensors mounted on the drone, which could compromise its agility.

Due to this, on most multirotors, the motors are controlled by a "throttle" command, which adjusts the duty cycle of a pulse-width modulation (PWM) signal applied to them. The actual rotational speed achieved by the motor depends not only on the throttle command but also on the battery voltage. [7].

Therefore, the formulation of the problem consists of finding a mapping function, called Thrust Map (TM), that relates the desired thrust command $T(N)$ to the throttle signal $Th(\%)$ required to achieve it, as shown in Equation 2.

*[1] Computer Vision and Aerial Robotics Group (CVAR), Centre for Automation and Robotics (C.A.R.), Universidad Politécnica de Madrid (UPM-CSIC), 28006 Madrid, Spain. Correspondence e-mail: fj.anguita@upm.es

[2] Automation and Robotics Research Group (ARG), Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, 1855 Luxembourg.

[1]Code repository: https://github.com/aerostack2/thrust_map_estimation

[2]Video: https://vimeo.com/1104217611/ce4e5d3f6c

$$Th = f(T, \ldots) \tag{2}$$

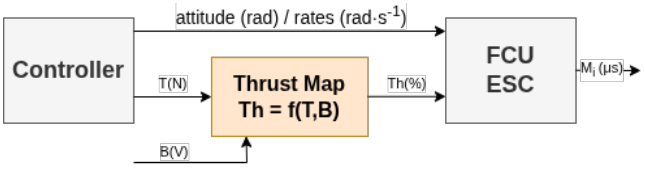


Figure 1: Block diagram of a typical thrust mapping system for quadrotor control. The low-level controller computes desired thrust values $T(N)$. These are converted into throttle commands Th$(\%)$ via a thrust map, which may depend on additional factors such as battery voltage $B(V)$. The throttle commands are then sent to the Flight Control Unit (FCU), which generates motor-specific PWM signals for actuation. These signals are transmitted to the Electronic Speed Controllers (ESCs), which regulate the power supplied to each motor, effectively controlling their speed.

Despite the importance of TM calibration, the field lacks a standardized approach to obtain and processing the data needed to build accurate maps. In this work, we focus on offline TM estimation to elaborate a detailed description of the TM calibration methods studied and report on their strengths and weaknesses based on experimental validation results. In summary, the main contributions of this work are:

1. A **comparative study** of multiple TM estimation strategies, including experimental measurements and in-flight refinements, highlighting trade-offs in accuracy and complexity.
2. A concise **methodology for offline TM estimation**, encompassing data collection, calibration, and evaluation steps.
3. A **correction procedure** that improves the accuracy of offline calibrated TMs by compensating for discrepancies observed during real flight operation.

2 RELATED WORK

TM characterization problem can be solved following two different approaches: offline calibration and online estimation. On the one hand, offline calibration refers to the process of characterizing the TM before the flight, using controlled and typically repeatable laboratory conditions. This is done by collecting thrust data through static tests (e.g., using a load cell or thrust stand) and then fitting a mathematical model—often a polynomial or surface function. On the other hand, online estimation involves determining or refining the TM during flight by using sensor data (e.g., IMU, state estimators) and real-time models. These methods dynamically adjust thrust estimates to better reflect current flight conditions, possibly adapting to changes in vehicle mass, battery level, or rotor wear.

The most extended method is the offline estimation because of its simplicity. In [8], a second-order polynomial curve is fitted to the data recorded from a motor mounted on a load cell. The work in [9] extends the 2D modeling of the TM by adding a linear dependency with the battery, thus creating a surface for the map. There are several techniques for obtaining the thrust data needed for this approach. These can be classified into direct experimental techniques, theoretical techniques, and simulation techniques.

The works in [6, 10] use commercial devices to take experimental measures of the thrust produced by the propeller. Other authors prefer to design custom thrust stands for this data recording [11]. There is, however, a general lack of description on how to process the obtained data. The work in [12] goes one step further by installing two motors onto their device in order to measure the effect of their simultaneous functioning on each other.

Regarding theoretical techniques, Classical Blade Element Theory (BET) and Blade Element Momentum Theory (BEMT) are used in [13] to obtain the force data for the map estimation. This theoretical approach is more difficult to apply to small UAVs due to aeroelastic effects and transverse flow, as described in [14].

Propeller simulation using Computational Fluid Dynamics (CFD) is an alternative to this kind of machinery. In [15, 16], CFD software is used to simulate the produced thrust data needed for the TM curves. The experimental data taken using a thrust stand is also compared to simulated CFD thrust data in [10]. However, this approach requires an accurate 3D model of the propeller, which is often not available.

No matter what method is used to gather the force or thrust measurements, it is important to note that most of the thrusts obtained using these methods are estimations, and even the experimental results may vary when all the rotors are mounted on the multirotor UAV. Regarding this matter, [8] proposes the use of a correction factor to address the difference between the estimated total thrust the multirotor will produce and the real thrust it is producing.

The problem of TM modeling can be tackled with online estimation and correction of the thrust commanded to the rotors. In [17], a BEMT-based estimator provides thrust observations for a feedforward thrust control scheme. Other techniques for online estimation of the quadrotor mass, like Least-Squares Estimation (LSE) or Extended Kalman Filter (EKF), are discussed in [18]. The thrust estimation among other parameters using a Bayesian filter is discussed in [19]. These approaches require a module to be running during flight time.

Despite the widespread use of offline TM calibration, we observe that existing work often lacks a comprehensive and reproducible methodology. In particular, there is limited guidance on how to design and conduct the necessary experiments or how to process the resulting data to obtain accurate and reliable TMs. This work addresses this gap by presenting a detailed calibration procedure that systematically covers

http://www.imavs.org/

http://www.imavs.org/

data acquisition, modeling, and evaluation, using various experimental sources.

In addition, most experimental setups used in the literature do not reproduce real flight conditions accurately. This mismatch can lead to TMs that do not generalize well to actual UAV operation. To address this, we introduce a custom extension to a commercial thrust stand to carry out experiments that better reproduce real flight conditions.

Lastly, even with careful offline calibration, discrepancies often remain between the predicted and actual thrust during flight. To mitigate this issue, we propose the addition of a correction factor in our procedure, similar to existing solutions in the literature, that refines the thrust estimation.

3 THRUST MAP CALIBRATION

3.1 Problem formulation

As mentioned in Section 1, the problem of TM calibration is the problem of approximating a function of the form described in Equation 2. This approximation can be carried out using different methods, including:

- Linear approximation.
- Polynomial approximation.
- Polynomial approximation with a correction factor.

The error committed by each method can be evaluated as the difference between the real throttle command sent during real operation, Th, and the throttle value predicted by the TM function, $\hat{Th}$, as shown in Equation 3:

$$e_{Th} = \left| Th - \hat{Th} \right| \tag{3}$$

3.2 Lineal approximation

Assuming a simplified model, the throttle signal Th can be considered a function of the desired thrust T as:

$$Th = f(T) \tag{4}$$

Under the assumption of a linear relationship between throttle and thrust, the TM function $\hat{f}$ can be approximated by a first-order expression:

$$f(T) = a \cdot T + b \tag{5}$$

where a and b are constants that define the slope and offset of the mapping. This model implies that when $T = 0$, the throttle is at its minimum value, and when $T = T_{\text{max}}$, the throttle reaches 100%.

In the particular case where the thrust-to-throttle ratio is constant and no offset is needed, the model can be further simplified as:

$$f(T) = \frac{1}{T_{\text{max}}} \cdot T \tag{6}$$

This simplification assumes that T_{max} is the maximum thrust generated when the throttle is fully applied (100%).

3.3 Polynomial approximation

To improve accuracy, the linear model can be extended by considering the influence of battery voltage on the available motor power. As the battery voltage drops, the motors' ability to generate thrust for a given throttle input decreases [20]. Therefore, a more realistic model considers both the desired thrust T and the current battery level B (in volts) as inputs.

$$Th = f(T, B) \tag{7}$$

In this work, the function f is approximated using data collected from experimental measurements, consisting of triplets of the form $Ds_i(T_i, B_i, Th_i)$ for $i = 1, \ldots, N$. These measurements represent the throttle value required to achieve a specific thrust at a given battery level. The procedure for collecting this dataset is described in Section 4.

To approximate f, a polynomial surface fitting approach is adopted. This model allows the throttle to vary smoothly with both thrust and battery level. The fitted function $\hat{f}$ is chosen from the space of bivariate polynomials of degree d, and is constructed by minimizing the total approximation error over all N data points in P.

$$\hat{f} = \underset{f \in \mathcal{P}_d}{\operatorname{argmin}} \sum_{i=1}^{N} |Th_i - f(T_i, B_i)| \tag{8}$$

Mathematically, the problem consists of finding the TM polynomial function $\hat{f}$ that minimizes the total fitting error.

3.4 Correction factor

The throttle value computed by the fitted TM function $\hat{f}(T, B)$ is responsible for generating the desired force through the drone's propulsion system. Ideally, if $\hat{f}$ were perfectly calibrated, the force produced by the drone would exactly match the thrust commanded by the controller.

However, as discussed in [8], discrepancies may arise between the expected thrust, predicted by the fitted model, and the actual thrust generated by the motors during flight. These discrepancies can arise due to in-flight effects.

To compensate for this issue, we introduce a correction factor γ. This dimensionless factor adjusts the thrust command before it is passed to the TM $\hat{f}$. The goal is to ensure that the final throttle output yields a force that matches more closely the intended thrust of the controller. This modified structure is depicted in Figure 2.

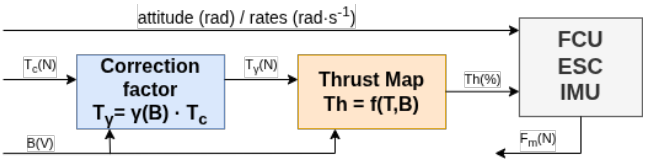


Figure 2: Diagram of the approach including the TM and the correction factor γ and the IMU that takes the acceleration measures to compute F_m, derived from Figure 1.

The corrected model is formulated as:

$$Th = \hat{f}(T_\gamma, B) = \hat{f}(\gamma(B) \cdot T_c, B) \tag{9}$$

For its computation, a dataset is collected using experimental measurements, consisting of triplets $Df_j(B_j, T_{c_j}, F_{m_j})$, for $j = 1, \ldots, N$, where T_{c_j} is the thrust command passed to the TM, and F_{m_j} is the measured z-force obtained, using the vertical acceleration of the drone a_z and mass m.

For each data point, the correction factor is estimated as:

$$\gamma_j = \frac{T_{c_j}}{F_{m_j}} \tag{10}$$

Then, a univariate polynomial of degree d is fitted to the data for solving the following minimization problem:

$$\hat{\gamma}(B) = \underset{\gamma \in \mathcal{P}_d}{\operatorname{argmin}} \sum_{j=1}^{N} \left| \frac{T_{c_j}}{F_{m_j}} - \gamma(B_j) \right| \tag{11}$$

This correction function $\hat{\gamma}(B)$ enables the system to dynamically adjust the input to the TM according to the battery level, ensuring that the resulting throttle command continues to produce a thrust consistent with the original intent of the controller.

4 EXPERIMENTAL VALIDATION

To obtain the triplets of data Ds and Df described in Section 3, experiments of two different natures are designed.

Firstly, thrust stand experiments are conducted to obtain the values for throttle commanded, thrust measured and battery level of triplet Ds, to estimate a TM using the Linear Approximation (LA) and the Polynomial Approximation. Using the Polynomial Approximation, two TM strategies will be followed: experiments with one rotor only will result in a Single Rotor (SR) TM, and experiments using all of the four rotors will be used to calibrate a Multirotor (MR) TM.

Then, using the SR and MR TMs, real flight experiments are carried out to obtain the Df triplets, with thrust commanded, battery level and z force measured, which allow us to compute a correction factor for these two TMs, obtaining the Single Rotor with correction factor (SR+γ) and Multirotor with correction factor (MR+γ).

Table 2 summarizes the types of experiments and the different TM approximations and strategies.

4.1 Experimental Setup

The TMs in this work are all calibrated for an xNova BlackThunder 2207-2100Kv racing motor connected to a Foxeer Reaper F4 65A ESC and powered by a TATTU R-LINE 6S LiPo battery. To obtain the data, these components are mounted on a commercial TYTO Robotics Series 1585 thrust stand. This device has three force gauges and an electronic board that measures the force and torque produced by the motor installed on it and stores the data, thus obtaining the Ds triplet to calibrate the SR TM. To perform four rotor experiments that better replicate real drone flight conditions, an extension is mounted and connected to the thrust stand signal output. This configuration allows to compute the MR TM with the new Ds triplet obtained. Figure 3 shows the full setup for the thrust stand experiments.

Experiment Type	TM Approximation	TM Strategy
Thrust Stand	Linear	LA
	Polynomial	SR
		MR
Flight Experiments	Polynomial + γ	SR+γ
		MR+γ

Table 2: Summary of data source experiments and their related TM strategies. The thrust stand experiments allow to compute a Linear, SR and MR approximated TMs. Flight Experiments allow to apply a correction factor γ to SR and MR TMs.

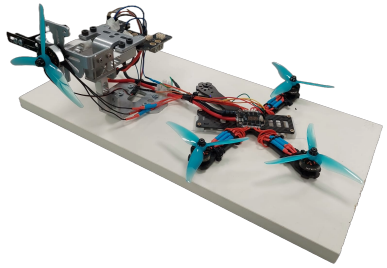


Figure 3: Commercial TYTO Robotics Series 1585 thrust stand and custom extension for multirotor experiments.

For real flight experiments, the same motors, ESC, and batteries were mounted on a 250 mm carbon fiber quadrotor frame. The ESC is controlled by a Foxeer H7 MPU6000 Flight Controller Unit (FCU). An Nvidia NX Orin is the onboard computer responsible for autonomous flying, and an Intel RealSense T265 camera is used for localization. The drone is shown in Figure 4.

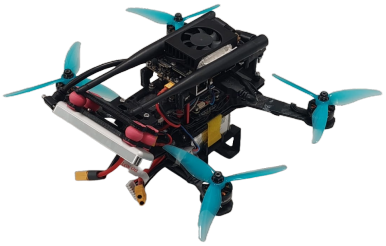


Figure 4: Autonomous drone used for real flight experiments.

In terms of autonomy, the computer onboard runs Aerostack2 [21]. This open-source framework performs the state estimation, computes the control, and sends the com-

http://www.imavs.org/

mands to the FCU to carry out specified tasks using its own mission planner. The controller used is a Model-Based Predictive Controller (MPC).

4.2 *Thrust stand experiments for TM calibration*

Series of three different types of experiments were carried out on the thrust stand in the form of steps and ramps (see Figure 5) that test both the transient and steady-state behavior of a single rotor.

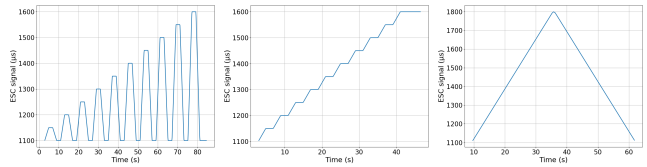


Figure 5: Sets of experiments conducted on the thrust stand to cover different operational points.

These same experiments were repeated with the four rotors connected to the ESC and commanded to rotate at the same speed and at the same time, recording a new set of data to which the new MR TM can be approximated.

With a simple experiment sending a maximum throttle command to the rotor, the maximum thrust that the rotor can produce is obtained, and thus the LA TM can be approximated using Equation 5.

Different polynomial surface alternatives are fitted to the data and tested using the error described in Equation 3 to find the surface that fits the best. Table 3 shows the error computed for each polynomial surface fit.

Polynomial	Error for SR	Error for MR
2nd degree	1.541 $\pm$ 1.936	1.009 $\pm$ 1.691
2nd deg. truncated	1.543 $\pm$ 1.956	1.041 $\pm$ 1.651
3rd degree	**1.411** $\pm$ **1.813**	**0.933** $\pm$ **1.390**
3rd deg. truncated	1.494 $\pm$ 1.886	1.009 $\pm$ 1.691

Table 3: Mean fitting error (%) computed for different degree polynomial surfaces fitted to the experimental data for SR and MR

Despite having a slightly lower error, the third degree polynomials overfit to the data for the throttle range that is being considered in this fitting process. A second degree polynomial is chosen as the best fit for the data in both cases, resulting in a TM function of the form:

$$Th = f(T, B) = a + b \cdot T + c \cdot B + d \cdot T^2 + e \cdot T \cdot B + f \cdot B^2 \quad (12)$$

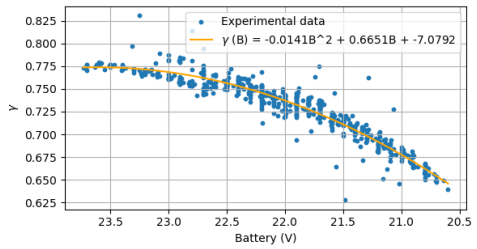
The fitted surfaces that characterize the SR TM and the MR TM are shown in Figure 6.

4.3 *Flight experiments for correction factor computing*

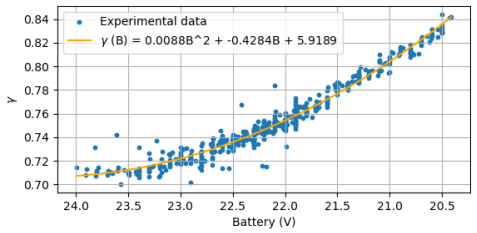
This factor was estimated off-board using data collected in D_f from two different flight experiments for each TM to be corrected (SR and MR). For each TM, two flights are performed: one long hover flight for the entire duration of the battery, and a flight with several short hover maneuvers at different heights. This way, distinct forces are required from the rotors, increasing the variety of the data. The same flight experiments are then performed to test the SR+γ and MR+γ TMs, as well as the performance of the SR and MR TM.

The measured force, $F_m = f(a_z, m)$, was computed using the z-axis acceleration measured by the IMU and the drone's mass. As expected, the experiments showed that even though the measured thrust must be constant while the drone hovers, the requested thrust increases as the battery level decreases.

Different datasheet of correction factors were computed for the various TMs. These factors were then fitted to the most suitable curve. The goal was to minimize the error (Equation 3) to find the best degree and form of the polynomial curve.



(a)



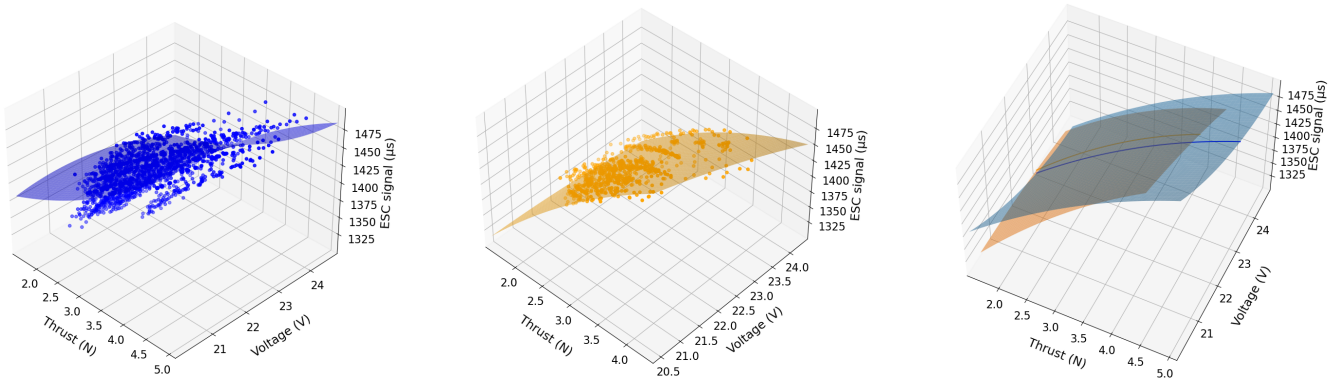
(b)

Figure 7: correction factor γ curves fitted to the data obtained from flight experiments. 7a shows the curve fitted to apply γ to the SR TM. 7b shows the curve fitted to apply γ to the MR TM.

Initially, a linear approximation dependency was used. However, this was not the best fit, so second- and third-degree approximations were tried. The results were very similar for both cases, so the second-degree polynomial approximation 13 was used to simplify the algorithm.

$$\gamma(B) = \gamma_0 + \gamma_1 \cdot B + \gamma_2 \cdot B^2 \quad (13)$$

http://www.imavs.org/



(a) (b) (c)

Figure 6: TM functions fitted from experimental data. 6a shows the polynomial fitted for SR data. 6b shows the polynomial fitted for MR data. 6c shows a comparative between SR (blue) and MR (orange) TMs.

http://www.imavs.org/

Figure 7 shows the performance of the polynomial fit chosen with the experimental data for the two TMs.

To validate these curves and ensure that the controller did not assume errors derived from the TM, the difference between the measured thrust and the commanded is calculated 14. The results gathered in Table 4 expose that $\gamma(B)$ absorbs the error e_{Th} from the TM and minimizes the error e_T.

$$\hat{e}_T = |T_c - T_m| = |T_c - f(a_z, m)| \tag{14}$$

Finally, the same flights were carried out with fitted correction factor curves added. The results are gathered in Table 4 and show in Figure 8, proving that the previous assumption that the error e_{Th} would be minimized with both TMs is correct.

TM Strategy	e_{Th} (%)	e_T (%)
LA	12.174 ± 1.417	0.538 ± 0.051
SR	4.370 ± 0.744	0.258 ± 0.031
MR	4.934 ± 0.675	0.235 ± 0.030
SR + $\gamma(B)$	0.557 ± 0.313	0.037 ± 0.015
MR + $\gamma(B)$	**0.197 ± 0.357**	**0.007 ± 0.006**

Table 4: e_{Th} and e_T computed for the different strategies. LA: linear approximation. SR: thrust stand with single rotor. MR: thrust stand with multi rotor. SR + $\gamma(B)$: thrust stand with single rotor and correction factor. MR + $\gamma(B)$: thrust stand with multi rotor and correction factor.

5 DISCUSSION

5.1 *Results from thrust stand experiments*

The thrust stand experiments results prove that there is a substantial difference between running the experiments exclusively on one rotor or connecting the other three to the ESC simultaneously. The ESC distributes the power from the battery to the rotors based on the throttle input it receives for each of the rotors. When only one rotor is connected, all the ESC power is dedicated to it. From experimental data, we know that the ESC inputs more than 20A of current to the rotor when it sends a full throttle command. The ESC used in these experiments works up to 65A, which is insufficient to send the needed power of a full throttle command to the four rotors at the same time. Besides, when the four rotors are connected, the same throttle command provokes a larger decay in the battery level, thus obtaining a lower thrust. These differences are shown in Figure 6. The surface corresponding to the MR TM in Figure 6c is at all times above the surface for the SR one. This means that for a given battery level, the MR TM estimates that a higher throttle command is needed to achieve a desired force. Whether this higher computed throttle is more accurate or not is proved in flight experiments.

5.2 *Results from Flight Experiments*

Flight experiments allow to analyze flight performance from the SR and MR TM computed using thrust stand experiments. Throttle and thrust errors from Figures 8 and 9 show that the MR TM has a poorer performance for higher battery levels, above 22.5V, than the SR TM, with a 5% throttle error against the 4% throttle error achieved by SR. Despite this, it has a lower error from 21.5V to the end of the battery duration, as MR throttle error goes down to 4% and SR error goes up to 6%. Looking at the evolution of the battery level in Figure 11, the battery level stays below 23.0V after time reaches 50 seconds, so for the biggest part of flight, the MR TM has a better performance than the SR TM. Even though the mean throttle error of 4.934% for MR TM shown in Table 4 is higher than the one for SR, which is 4.370%, flight experiments show that the MR TM performs a better hover. In Figure 10 it is shown that once the battery level reaches the range of better MR performance, the drone begins to approach the 3 meter reference to hover and improves its flight. However, the SR TM starts to diverge more and more from this reference as the battery runs lower.

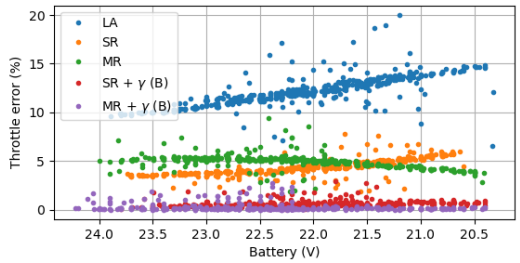


Figure 8: e_{Th} computed for the different TM strategies along the battery range.

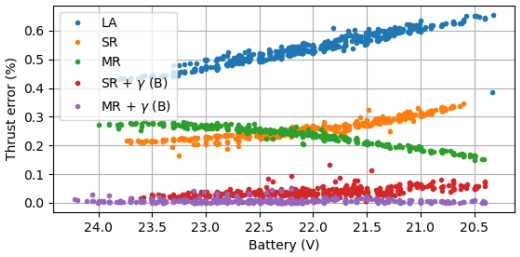


Figure 9: e_T computed for the different TM strategies along the battery range.

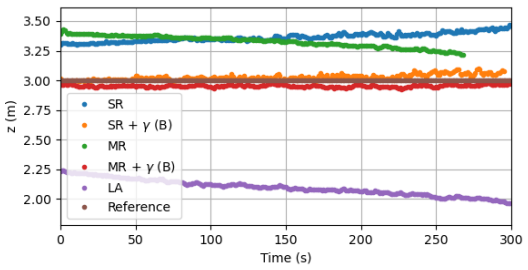


Figure 10: Drone stability performance when commanded to hover at 3 meters height reference for each TM.

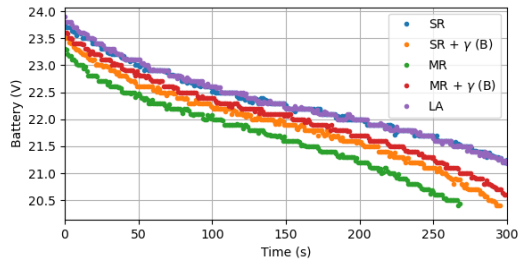


Figure 11: Battery level over time for different strategies

The results for the linear approximation highlight the importance of calibrating a proper TM. This simple solution has an error 10-20 times higher than the other approaches (see Figure 8), and even though it is still possible to fly with such a TM, the trajectory tracking error in the z-axis and the ability to keep hover height are quite poor (see Figure 10).

The addition of a correction factor to previously calibrated TMs is a round success. The throttle error is reduced from 12% to almost 0.5%, as can be seen in Table 4, and the hover error stays within 5 centimeters for a hover reference of 3 m (2%), as shown in Figure 10, throughout the battery range. This error reduction is even higher for the MR+γ TM, achieving a throttle error below 0.2% (see Table 4). This is also due to the MR+γ TM spending more time operating in its most favorable battery range, as previously commented. Despite having lower throttle and thrust errors than SR+γ, the hover performance is very similar, with a hover error also below 2%.

5.3 Comparative of TMs

SR and MR error and flight performance can be improved considerably by conducting a couple of flight experiments with which to compute a correction factor that reduces that error to less than 1%. The MR TM has not proven to be as useful as the corrected counterparts. Its mean throttle error increases with respect to the SR one when no correction factor is applied. Figure 6b shows that fewer data were taken from the four rotor experiments, due to the faster discharge of the batteries. This could have affected the accuracy of the MR TM computing. More experiments and more data might result in a better calibration of this MR TM. Even though MR TM shows a slightly better overall hover performance than SR in Figure 10, we consider this improvement not worth the effort to calculate an MR TM alone. When the correction factor was added to the MR TM, the error obtained for the MR + γ proves to be lower than the SR + γ, going down to 0.197%. Although this reduction in the error is significant, the SR+γ shows a good flight performance, with a hover error also within 5cm for a 3 meter hover flight and a throttle error below 1%. We consider the SR+γ TM to be the better strategy in a performance/computation cost relation. TM performance can be slightly improved if MR+γ is a possibility, i.e. if mounting resources and a safe space for experiments are available.

6 Conclusion

In this work, we proposed a methodology for Thrust Map estimation that proves to be effective, allowing the estimation of different TMs that predict the forces the drone will produce in real flight experiments.

We also presented an extension to the general approach using a correction factor that is computed from flight data. By applying this correction factor, great improvement is achieved for the in real flight experiments.

Based on these same flight experiments, a comparison of TM calibration strategies was performed, and a TM alternative is proposed as the best overall approach: a Single Rotor with Correction Factor (SR+γ) Thrust Map.

As a proposal for future work, additional TMs can be calibrated using different ESCs and batteries to compare how

http://www.imavs.org/

these components affect the TM for the same set of motors. A change in the motors would undeniably call for a new TM, but the dependency on the selected ESCs and batteries is not so evident.

Additionally, considering how similar the process of fitting a TM or a correction factor to data and computing the error is to the training of an AI agent, these polynomial curves and surfaces fitting can be substituted by machine learning techniques.

ACKNOWLEDGEMENTS

We acknowledge the support of the European Union through the Horizon Europe Project No. HZ230070443 SHEREC: Safe, Healthy and Environmental Ship Recycling and the Horizon Europe Project No. 101070254 CORESENSE. This work has also been supported by Grant ref. ID2021-127648OBC32 INSERTION funded by MICIU/AEI/10.13039/501100011033 and by FEDER, EU. In addition, this work has also been supported by the project RATEC ref: PDC2022-133643-C22 funded by the Spanish Ministry of Science and Innovation. The work of the second author is supported by the Pre-Doctoral Program of Research and Technologic Innovation Department of Madrid Government. The work of the fourth author was funded by the Luxembourg National Research Fund (FNR) under the DEUS Project (Ref.C22/IS/17387634/DEUS).

REFERENCES

[1] Muhammad Maaruf, Magdi Sadek Mahmoud, and Alfian Ma'arif. A survey of control methods for quadrotor uav. *International Journal of Robotics and Control Systems*, 2(4):652–665, 2022.

[2] Yu Wang, Kang Liu, Peiyin Chen, Huiliao Yang, and Di Wang. Adaptive fuzzy fixed-time tracking control for maritime quadrotor uav landing onto a moving usv with deck motion prediction. *Nonlinear Dynamics*, 113(13):16857–16876, 2025.

[3] Paulo Jefferson Dias de Oliveira Evald, Vivian Misaki Aoki, César Bastos da Silva, Dayana Santos Cardoso, Pedro Miranda Pinheiro, Silvia Silva da Costa Botelho, and Paulo Lilles Jorge Drews Junior. A review on quadrotor attitude control strategies. *International Journal of Intelligent Robotics and Applications*, 8(1):230–250, 2024.

[4] Burak Yüksel, Cristian Secchi, Heinrich H Bülthoff, and Antonio Franchi. A nonlinear force observer for quadrotors and application to physical interactive tasks. In *2014 IEEE/ASME international conference on advanced intelligent mechatronics*, pages 433–440. IEEE, 2014.

[5] Bowen Zhang, Zaixin Song, Fei Zhao, and Chunhua Liu. Overview of propulsion systems for unmanned aerial vehicles. *Energies*, 15(2):455, 2022.

[6] Riccardo Spica, Paolo Robuffo Giordano, Markus Ryll, Heinrich H Bülthoff, and Antonio Franchi. An open-source hardware/software architecture for quadrotor uavs. *IFAC Proceedings Volumes*, 46(30):198–205, 2013.

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

[8] Matthias Faessler, Davide Falanga, and Davide Scaramuzza. Thrust mixing, saturation, and body-rate control for accurate aggressive quadrotor flight. *IEEE Robotics and Automation Letters*, 2(2):476–482, 2016.

[9] Daniel Hentzen, Thomas Stastny, Roland Siegwart, and Roland Brockers. Disturbance estimation and rejection for high-precision multirotor position control. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2797–2804. IEEE, 2019.

[10] Zaid Siddiqi and Jin Wook Lee. A computational and experimental study on aerodynamics of motor-driven propellers using thrust stand and rotating cup anemometer. *Progress in Computational Fluid Dynamics, an International Journal*, 22(1):23–36, 2022.

[11] Marián Kišev, Lukáš Vacho, Ladislav Tóth, Martin Olejár, Marta Harničárová, Jan Valíček, Hakan Tozan, et al. Experimental measurement of a uav propeller's thrust. *Tehnički vjesnik*, 29(1):73–80, 2022.

[12] Bart Theys, Grigorios Dimitriadis, Patrick Hendrick, and Joris De Schutter. Influence of propeller configuration on propulsion system efficiency of multi-rotor unmanned aerial vehicles. In *2016 international conference on unmanned aircraft systems (ICUAS)*, pages 195–201. IEEE, 2016.

[13] Rajan Gill and Raffaello D'andrea. Propeller thrust and drag in forward flight. In *2017 IEEE Conference on control technology and applications (CCTA)*, pages 73–79. IEEE, 2017.

[14] Zbigniew Czyż, Paweł Karpiński, Krzysztof Skiba, and Mirosław Wendeker. Wind tunnel performance tests of the propellers with different pitch for the electric propulsion system. *Sensors*, 22(1):2, 2021.

[15] Alexander H Bryant and Dibbon K Walters. Developing a small-scale propeller thrust model using experimentation and cfd. In *AIAA Scitech 2021 Forum*, page 1307, 2021.

[16] C Paz, E Suárez, C Gil, and J Vence. Assessment of the methodology for the cfd simulation of the flight of a quadcopter uav. *Journal of Wind Engineering and Industrial Aerodynamics*, 218:104776, 2021.

[17] Davi Santos, Martin Saska, and Tiago Nascimento. A generalized thrust estimation and control approach for multirotors micro aerial vehicles. *IEEE Robotics and Automation Letters*, 2024.

[18] Du Ho, Jonas Linder, Gustaf Hendeby, and Martin Enqvist. Mass estimation of a quadcopter using imu data. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1260–1266. IEEE, 2017.

[19] Christoph Böhm, Martin Scheiber, and Stephan Weiss. Filter-based online system-parameter estimation for multicopter uavs. In *Robotics: Science and Systems*, 2021.

[20] Leonard Bauersfeld and Davide Scaramuzza. Range, endurance, and optimal speed estimates for multicopters. *IEEE Robotics and Automation Letters*, 7(2):2953–2960, 2022.

[21] Miguel Fernandez-Cortizas, Martin Molina, Pedro Arias-Perez, Rafael Perez-Segui, David Perez-Saura, and Pascual Campoy. Aerostack2: A software framework for developing multi-robot aerial systems. *arXiv preprint arXiv:2303.18237*, 2023.

http://www.imavs.org/

A Comparison of Optimised, Fixed-Tilt, Fully-Actuated Multirotor UAV Configurations

S. Al-zubaidi and K. A. Stol
Department of Mechanical and Mechatronics Engineering,
University of Auckland, New Zealand

http://www.imavs.org/

ABSTRACT

Multirotor Unmanned Aerial Vehicles (UAVs) that are fully-actuated offer improved dexterity for aerial manipulation tasks by decoupling attitude and translation. There is also potential for improved agility when the airframe design is optimised, which can lead to better disturbance rejection. This paper compares three fixed-tilt configurations by evaluating optimal designs of each as a function of different design requirements: payload, flight time, and horizontal force while in level hover. Three different characteristics are used as objective functions in the optimisation: thrust bandwidth, airframe mass, and airframe diameter. The results highlight the advantages of a heterogeneous rotor configuration, providing the best trade-off between agility and energy efficiency.

1 Introduction

In recent years, novel multirotor Unmanned Aerial Vehicle (UAV) configurations have been introduced that are fully-actuated [1-9], driven mainly by aerial manipulation of physical interaction tasks [10, 11]. Examples of fully-actuated configurations are shown in Figure 1. Some designs satisfy a specific operational requirement, such as the placement of a tool requiring a large clearance between two sets of rotors [5]. However, in most cases, the rotor configurations are not designed with an application in mind. Across all the possible configurations, it is not clear which has the potential to operate most favourably as design requirements are changed.

Full-actuation in multirotor UAVs can be obtained via use of fixed-tilt rotors [1-3, 12-15], variable-tilt rotors [16-19] or variable-pitch rotors [20, 21]. Fixed-tilt rotors are mechanically the simplest but sacrifice flight endurance and are the focus of this paper. Different configurations of fixed-tilt rotors are possible. The simplest is to tilt all rotors to the same angle about either one [2, 3, 12, 13] or two [1, 14, 15] axes. One of our studies [1] concluded that there is no advantage to tilting rotors about an axis perpendicular to the rotor arm, also known as dihedral. Having symmetry and uniformity of rotor tilt is helpful in creating a structurally simple system. However, having different tilt for each rotor allows for either the use of fewer rotors [9], omnidirectional motion [4], or unequal force capabilities [6]. In these cases, there is a clear link between the configuration and the performance advantages they may provide.

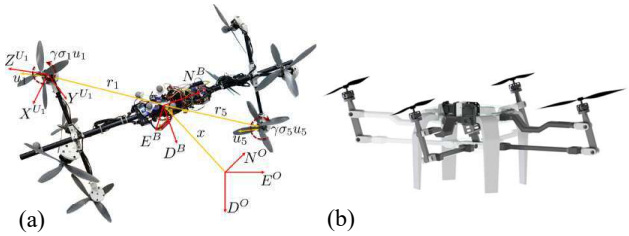


Figure 1: Examples of fully-actuated multirotor UAVs: (a) fixed-tilt octorotor [5], (b) variable-tilt quadrotor [19].

A clear link between rotor configuration and performance is not always the case. For example, consider heterogeneous configurations, which contain different sets of rotors [1, 7, 23]. In our research group's earlier work, heterogeneous configurations were introduced to solve the problem of the decreased agility of homogenous configurations at high payload mass [2] and, in [1] were shown to have a higher thrust bandwidth at one design condition. This advantage did not hold for other design conditions, especially at lower payload mass, where the homogenous configuration performed well [3]. We explored the use of coaxial and overlapping rotors, involving either all rotors [2] or a subset of the rotors [1]. However, a direct comparison was not made to a configuration with non-coaxial rotors.

Previous studies have attempted to quantify agility. One measure is the bandwidth of the rotor thrust response horizontally [1, 2]. Another is the maximum rotor thrust [4]. In [25], the author introduced nine different experimental ways to measure the agility of a UAV. An example of the introduced methods is the maximum attitude acceleration and time needed to reach a particular attitude. The author in [3] performed an optimisation where horizontal force has a minimum requirement and assumed that after optimisation, the UAV would have a maximum force equal to the required force; thus, rather than using acceleration, the mass of the UAV was used to characterise agility.

The main contributions of this paper are (1) an approach that fairly compares fully-actuated airframes using optimal solutions of each with identical performance requirements, and (2) a thorough comparison of three octorotor configurations. The comparison is performed across different design requirements and three different objective functions to illustrate how the approach can identify the most favourable configuration.

The remainder of this paper is organised as follows. Section 2 describes the case study undertaken. The results for the different optimisations are presented in sections 3, 4, and 5. Section 6 discusses the implications of the results. The paper is concluded in Section 7.

http://www.imavs.org/

2 Description of Case Studies

This section describes the case studies performed, including the different configurations, design requirements and objective functions used. Optimisations use the aerodynamic, electrical, and structural models described in [22] except that the calculation of rotor moment of inertia uses the improved model in [26].

2.1 Configurations

Three different configurations with eight fixed-tilt rotors are considered: *Planetary Hex* [1], *Stacked Octo* [2] and *Planar Octo* [3], illustrated in Figure 2. These configurations are chosen because all were previously optimised for their agility. Both the Planar Octo and Stacked Octo configurations are each considered to be homogenous configurations due to having rotors of the same diameter and tilt angles. The Planetary Hex is a heterogenous configuration, using two different sets of rotors: 'sun' rotors in the centre and 'planet' rotors around the perimeter. The Planetary Hex and Stacked Octo configurations have coaxial or overlapping rotors with a thrust penalty of 20% applied to the lower rotors [24].

2.2 Optimisation Problem

A constrained optimisation problem consists of constraints, referred to here as *design requirements*, and the *objective function*. The design requirements chosen are the payload mass, hover flight time, and desired horizontal force. The payload is defined as all non-flight-critical components, such as sensors, companion computers, and manipulators. The horizontal force is defined as the magnitude of a persistent force that the UAV rotors must produce in a horizontal plane while the aircraft is hovering in level flight. The force requirement could, for example, be dictated by an aerial manipulation task or aerodynamic drag from a steady wind. Table I lists the nominal values of the design requirements and the ranges over which each were varied. Within the case studies, one design requirement is changed at a time. For example, when varying the payload from 1 kg to 20 kg, the required hover time and horizontal force are 10 mins and 2 N, respectively.

For the optimisation objective function, three different airframe characteristics are explored separately: (1) open-loop bandwidth of rotor thrust, (2) airframe mass, and (3) airframe diameter. The importance of each characteristic would depend on what the designer deems the most important performance objective. If the UAV is intended to station-keep while rejecting wind disturbances, the rate at which the rotors can adjust thrust is important, correlated to high thrust bandwidth. If the UAV is required to track a fast-changing trajectory, the airframe mass should be minimised. If the UAV must operate in confined spaces, then a small airframe diameter is required.

Requirement	Nominal	Range
Payload	1 kg	1 – 20 kg
Hover Time	10 min	5 – 30 min
Horizontal Force	2 N	2 – 20 N

Table I: Design requirements: nominal and sweep range.

2.3 Optimisation Method

In our previous work [22], we describe a two-stage hybrid optimisation method to deal with the continuous variables (such as rotor tilt angles) and discrete variables (motor and propeller components). In the first stage, a surrogate optimisation finds an initial design by using empirical models

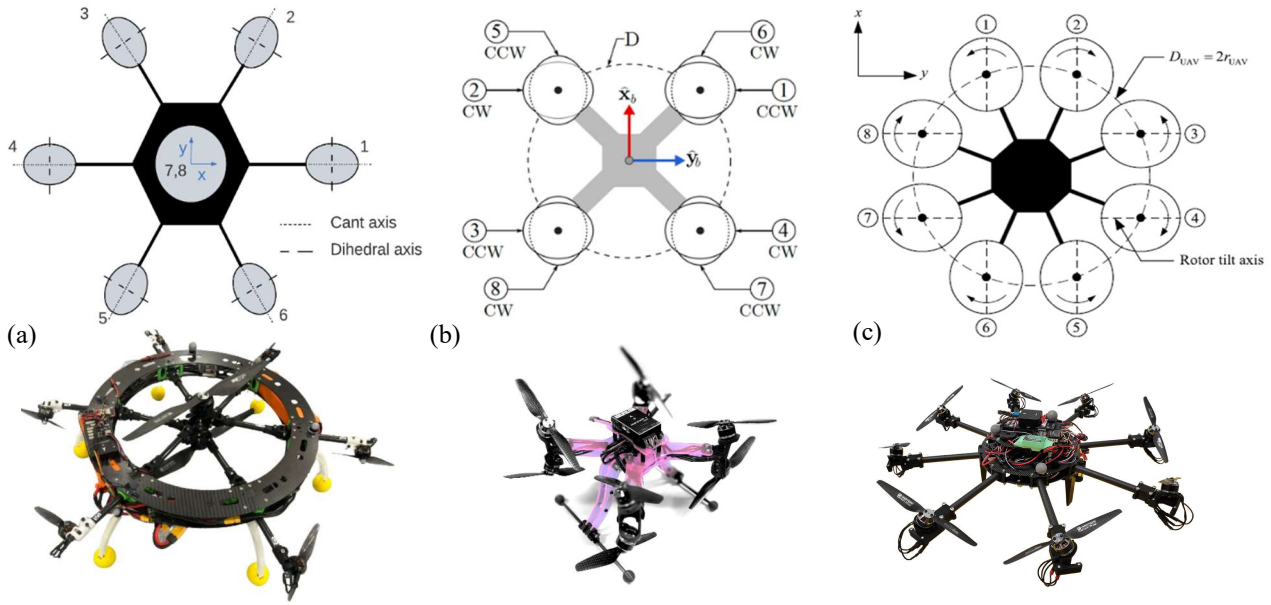


Figure 2: Rotor configurations examined and flight-tested instances of each (not to scale):
(a) Planetary Hex [1], (b) Stacked Octo [2], (c) Planar Octo [3].

http://www.imavs.org/

for the relationships between the discrete design variables, simplifying the problem to one involving only continuous variables. The empirical models are based on a database of 250 motors and 84 propellers. The second stage is a localised parameter sweep of only discrete design variables, arriving at a design that can be fabricated. In this paper, we examine the optimal solutions after only the initial stage so that trends are not dependent on the discrete components available.

The set of continuous design variables, X, for optimising each configuration are listed in Table II. All other geometric properties are dependent on these variables. The additional variable, *planet hover thrust*, specifies the proportion of the total vertical thrust that is contributed by the planet rotors.

Planetary Hex	**Planar Octo**	**Stacked Octo**
Planet rotor tilt angle	Rotor tilt angle	Rotor tilt angle
Sun rotor diameter Planet rotor diameter	Rotor diameter	Rotor diameter
Planet hover thrust		

Table II: Design variable sets, X, for each configuration.

3 Maximum Thrust Bandwidth

This set of case studies considers horizontal thrust bandwidth, f_{HT}, requiring the optimisation objective function

$$\max_X f_{HT}. \tag{1}$$

3.1 Effect of Changing the Payload Requirement

In this subsection, optimal designs are calculated for different payload mass requirements while keeping hover time and horizonal force constant at 10 minutes and 5 N, respectively. The resulting thrust bandwidth is shown in Figure 3. The small trend fluctuations are likely caused by the battery model requiring an integer number of LiPo cells.

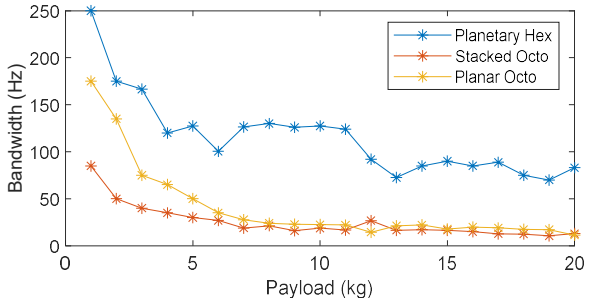


Figure 3: Effect of changing the payload mass on thrust bandwidth.

As payload increases, the thrust bandwidth decreases for each configuration. This is an expected result due to needing larger rotors to lift larger mass UAVs. Larger rotors have a higher thrust bandwidth due to their larger mass moment of inertia. Previous work has demonstrated the superior bandwidth of the Planetary Hex configuration at a single low payload [1]. Figure 3 demonstrates superiority across all payloads, by as much as 5 times for payloads greater than 7 kg. The Planar Octo and Stacked Octo configurations perform similarly at the higher payloads. At lower payloads, the differences in bandwidth are more pronounced due the more significant impact of the shadowed lower rotors on the Stacked Octo.

3.2 Effect of Changing the Hover Time Requirement

In this subsection, only the hover time requirement is varied, while payload and required horizontal force are kept constant. Results are shown in 4. The decreasing bandwidth with flight time for all configurations is logical because proportionally larger battery capacities are required, as shown in Figure 5. This leads to increased battery mass, total mass, and larger rotors, consistent with the payload study.

Comparing configurations, there are no unexpected trends with thrust bandwidth. The Planetary Hex is superior, followed by the Stacked Octo and Planar Octo. The Planar Octo requires the lowest battery capacity and consequently the lowest current draw for a given flight time.

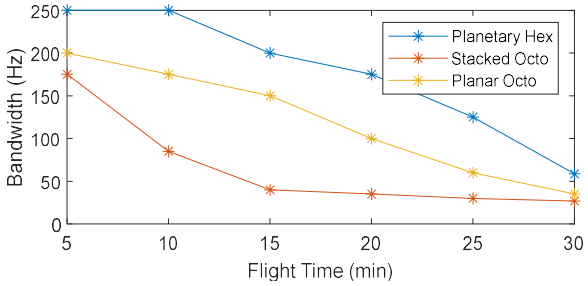


Figure 4: Effect of changing hover time on thrust bandwidth.

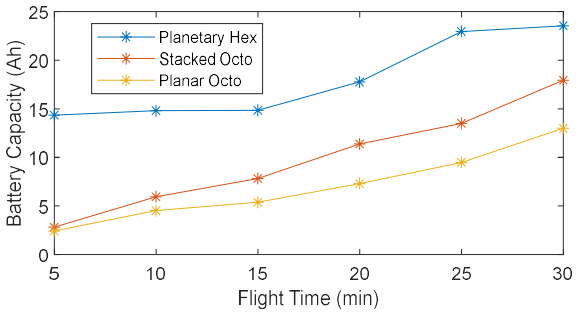


Figure 5: Effect of hover time on required battery capacity.

3.3 Effect of Changing the Horizontal Force Requirement

In this subsection, only the horizontal force requirement is varied, while payload and hover time are kept constant. The results are presented in Figure 6. Larger horizontal force is achieved by a combination of larger tilt angles and larger rotor diameters, resulting in lower thrust bandwidth, regardless of the configuration. For example, the trend in planet tilt angles for the Planetary Hex is shown in Figure 7.

The Planetary Hex configuration maintains a higher bandwidth across the entire range of required horizontal force, followed by the Planar Octo and Stacked Octo configurations.

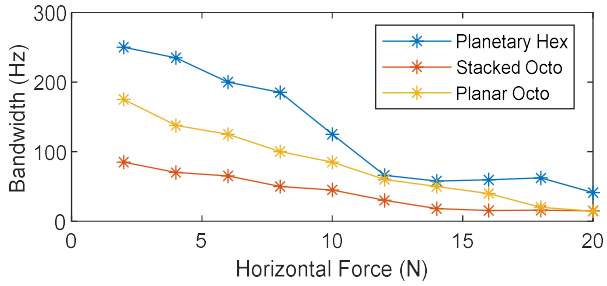


Figure 6: Effect of changing the horizontal force requirement on thrust bandwidth.

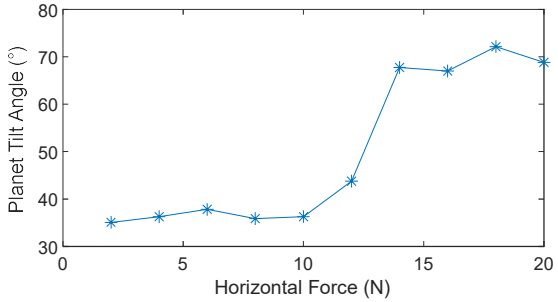


Figure 7: Effect of changing the horizontal force requirement on planet rotor tilt.

4 Minimum Mass

This set of case studies considers optimising for lowest mass. The mass of the payload, m_{pay}, is subtracted from the total mass of the UAV, m_{UAV}, to isolate the airframes in the comparisons. The optimisation objective function is thus

$$\min_{X}\left(m_{UAV} - m_{pay}\right) . \quad (2)$$

4.1 Effect of Changing the Payload Requirement

This case study explores the effect of increasing the payload on the bandwidth of the optimal UAV of each of the three configurations. Linear trends can be seen in Figure 8. As expected, with an increase in payload, the mass of the UAV also increases. Higher thrust demands with payload increase the required battery capacity, rotor sizes, and frame sizes, all of which increase UAV mass.

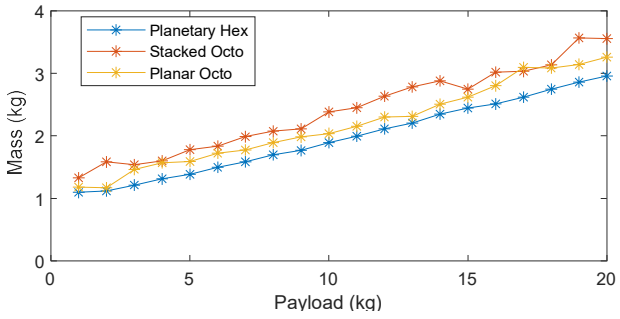


Figure 8: Effect of changing the payload mass on UAV mass (excluding payload).

The Stacked Octo and Planar Octo have very similar mass airframes across all payloads tested. Marginally, the Planetary Hex is consistently the lightest configuration. Examining the mass of the different sets of components that comprise the UAV airframe, Figure 9, reveals that the Planetary Hex can have the lowest mass because of its heterogeneous set of rotors. Effectively, having 2 larger rotors and 6 small rotors is generally lighter than 8 rotors of the same size.

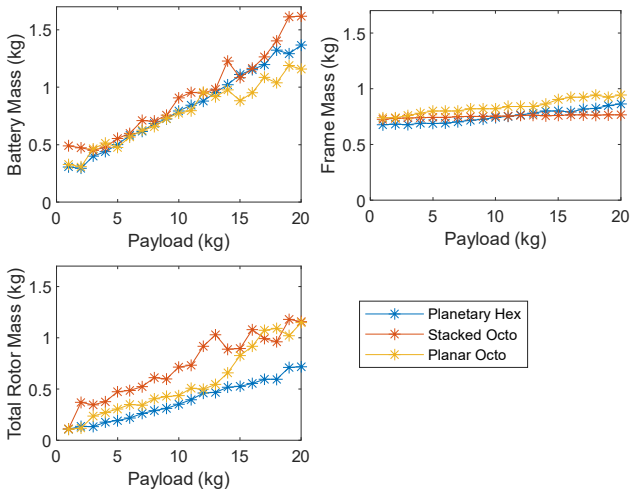


Figure 9: Mass of individual UAV components.

4.2 Effect of Changing the Horizontal Force Requirement

This case study explores the effect of increasing the horizontal force requirements on the mass of the optimal UAV for each of the three configurations. The increase in force requirements is expected to increase the mass for each configuration. This is because the tilt angle will have to be increased, resulting in a larger battery capacity and mass. As seen in Figure 10, the mass of the Planetary Hex configuration is lower across all horizontal force requirements than the other two configurations. In addition, as in the previous case studies, the Stacked Octo is inferior to the other two configurations.

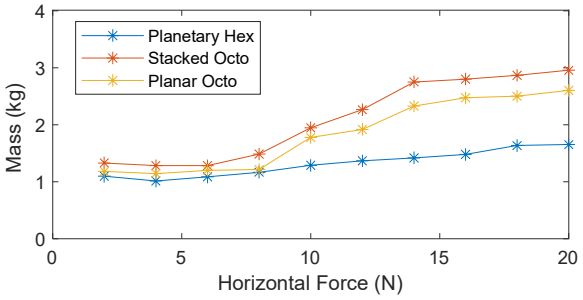


Figure 10: Effect of changing the horizontal force requirement on UAV Mass (excluding payload).

A significant trend in the Planetary Hex design as horizontal force requirement increases is the angle of the planet rotors, as shown in Figure 11. Increasing the tilt angle allows more thrust within the horizontal plane, increasing the

http://www.imavs.org/

horizontal force capabilities. When not required, a low angle is favourable as it means less opposing thrust, i.e. more efficient UAV and lower battery capacity required, resulting in a lower mass.

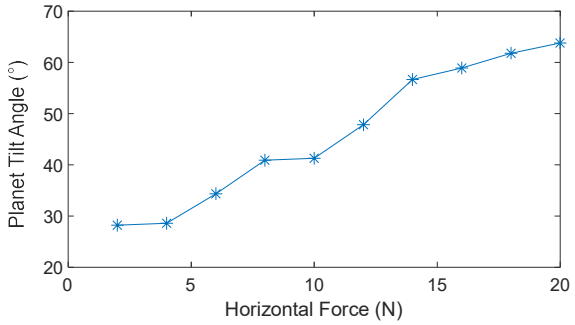


Figure 11: Effect of changing horizontal force requirement on planet rotor tilt.

5 Minimum Diameter

This case study involves minimising the diameter of the UAV, D_{UAV} and examines the effect of changing only one design requirement, the payload mass. Here, diameter of the UAV is defined as the diameter of the circle passing through the centres of each outer rotor. The diameter is expected to increase with the payload because of the need for larger rotors to produce more lift. The results in Figure 12 confirms this expected trend.

Up to payloads of 5 kg, there is no significant difference between the minimum diameters of all three configurations. Above 5 kg payload, the Planetary Hex is consistently the smallest configuration, followed by the Stacked Octo, and then the Planar Octo. It follows that configurations with rotors that are stacked or coaxial can achieve a smaller diameter. The losses caused by rotor overlap are not significant when diameter is the optimisation objective.

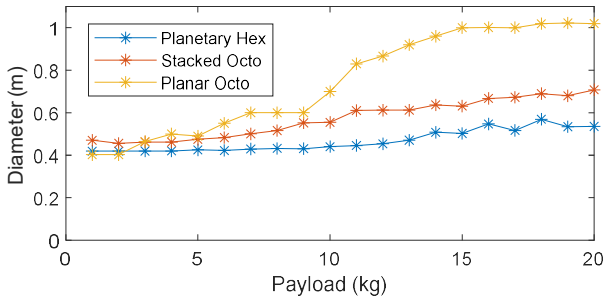


Figure 12: Effect of changing payload mass on UAV diameter.

6 Discussion

Some common findings are observed across the different case studies. Firstly, the heterogeneity of the Planetary Hex was a significant reason behind the configuration being superior compared to the other two under all situations. Heterogeneity contributes to this improvement through two main methods. The first aspect is evident when the two rotor sets are responsible for different design requirements. An example is the bandwidth case study with the payload requirement change. For a significant range of the payloads tested, it was found that the planet rotors, which are responsible for the thrust bandwidth, did not change, but the sun rotors adjusted to produce a UAV that could satisfy the requirements.

The other method in which heterogeneity contributes to improved design is the ability to reduce UAV capabilities without affecting the requirements. In some instances, using the same eight rotors compared to 2 different rotor sets totalling eight rotors is not advantageous as both achieve the same design requirements. The heterogeneous UAV can use a lower-performing rotor as it does not contribute; it will only require a low contribution to achieve the UAV's design requirements. This condition is demonstrated in the mass case study with the horizontal force requirement change. It was found that the sun rotor used was selected within certain design regions for its light mass, as it only needed to contribute a small amount of the total thrust for hover. Similarly, this effect is clear when looking at the change in planet rotors.

The other modification is the use of overlapping rotors, either coaxial or stacked. Using overlapping rotors was consistently unfavourable except when optimising for smallest diameter. In a case where a UAV hovers far away from obstacles, size would not be a major factor.

7 Conclusions

The paper presented an approach to fairly compare design characteristics of different UAV configurations by ensuring each design is optimised with the same design constraints. Using this approach, studies were presented to evaluate the thrust bandwidth, mass, and diameter by comparing three UAV configurations: Planetary Hex, Planar Octo, and Stacked Octo. The Planetary Hex proved superior in all cases due to its heterogeneous rotor design. Of the homogeneous configurations, the Stacked Octo was inferior when optimising for bandwidth or mass but superior to the Planar Octo when optimising for minimum diameter.

The results are based on empirical models and make several simplifying assumptions. Experimental work is required to validate the comparisons, such as by building and bench-testing a sample of the optimal designs. This study could be further expanded by including other fully-actuated UAV configurations.

Acknowledgement

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.

http://www.imavs.org/

http://www.imavs.org/

REFERENCES

[1] Al-Zubaidi, S., Stol, K., "Preliminary design optimisation of a novel fixed-tilt heterogeneous UAV for horizontal agility," *International Conference on Unmanned Aircraft Systems (ICUAS),* June 2022, Dubrovnik, Croatia, pp. 1489–1496.

[2] Souza, P.H.M., Stol, K., "Analysis and Design of a Novel Fully-Actuated Compact Multirotor," *Australasian Conference on Robotics and Automation (ACRA),* December 2021, Melbourne, Australia.

[3] Chen, Z.J., Stol, K., Richards, P.J., "Preliminary design of multirotor UAVs with tilted-rotors for improved disturbance rejection capability," *Aerospace Science and Technology,* Vol. 92, 2019, pp. 635–643.

[4] Brescianini, D., D'Andrea, R., "An omni-directional multirotor vehicle," *Mechatronics,* Vol. 55, 2018, pp. 76–93.

[5] Sangyul Park, Jeongseob Lee, Joonmo Ahn, "ODAR: Aerial Manipulation Platform Enabling Omnidirectional Wrench Generation," *IEEE Transactions on Mechatronics,* Vol. 23, No. 4, 2018, pp. 1907–1918.

[6] Nikou, A., Gavridis, G.C., Kyriakopoulos, K.J., "Mechanical design, modelling and control of a novel aerial manipulator," *IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 4698–4703.

[7] J. Verbeke, D. Hulens, H. Ramon, "The design and construction of a high endurance hexacopter suited for narrow corridors," *International Conference on Unmanned Aircraft Systems (ICUAS),* 2014, pp. 543–551.

[8] Tognon, M., Franchi, A., "Omnidirectional Aerial Vehicles with Unidirectional Thrusters: Theory, Optimal Design, and Control," *IEEE Robotics and Automation Letters,* Vol. 3, No. 3, 2018, pp. 2277–2282.

[9] Hamandi, M., Sawant, K., Tognon, M., "Omni-Plus-Seven (O 7 +): An Omnidirectional Aerial Prototype with a Minimal Number of Uni-directional Thrusters," *International Conference on Unmanned Aircraft Systems (ICUAS)*, Oct 2020.

[10] Telli, K., Kraa, O., Himeur, Y., "A Comprehensive Review of Recent Research Trends on Unmanned Aerial Vehicles (UAVs)," *Systems,* Vol. 11, No. 8, 2023,

[11] Sanchez-Cuevas, P.J., Gonzalez-Morgado, A., Cortes, N., "Fully-Actuated Aerial Manipulator for Infrastructure Contact Inspection: Design, Modeling, Localization, and Control," *Sensors,* Vol. 20, No. 17, 2020, pp. 4708.

[12] Ryll, M., Muscio, G., Pierri, F., "6D interaction control with aerial robots: The flying end-effector paradigm," *The International Journal of Robotics Research,* Vol. 38, No. 9, 2019, pp. 1045–1062.

[13] Jiang, G., Voyles, R., "Hexrotor UAV platform enabling dextrous interaction with structures-flight test," *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Oct 2013, pp. 1–6.

[14] Jiang, G., Voyles, R., Sebesta, K., "Estimation and optimization of fully-actuated multirotor platform with nonparallel actuation mechanism," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep 2017, pp. 6843–6848.

[15] Jiang, G., Voyles, R.M., Choi, J.J., "Precision Fully-Actuated UAV for Visual and Physical Inspection of Structures for Nuclear Decommissioning and Search and Rescue," *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Aug 2018, pp. 1–7.

[16] Junaid, A., Sanchez, A., Bosch, J., "Design and Implementation of a Dual-Axis Tilting Quadcopter," *Robotics,* Vol. 7, No. 4, 2018, pp. 65.

[17] Ding, C., Lu, L., Wang, C., "Modelling and Control of Fully Actuated Vector Thrust Unmanned Aerial Vehicles," *International Symposium on Flexible Automation,* July 2018, pp. 451–458.

[18] Segui-Gasco, P., Segui-Gasco, P., Al-Rihani, Y., "A Novel Actuation Concept for a Multi Rotor UAV," *Journal of Intelligent & Robotic Systems,* Vol. 74, No. 1, 2014, pp. 173–191.

[19] Taylor, J., Edwards, M., Stol, K., "Assessing the Agility of a Variable-Tilt UAV," *International Conference of Micro Aerial Vehicles (IMAV)*, Sep 2023.

[20] Gao, L., Zhao, J., Zhu, Y., "Application of Cycle Variable Pitch Propeller to Morphing Unmanned Aerial Vehicles," IEEE International Conference on Information and Automation, August 2015, pp. 2493–2498.

[21] Paulos, J., Yim, M., "Cyclic Blade Pitch Control Without a Swashplate for Small Helicopters," *Journal of Guidance, Control, and Dynamics,* Vol. 41, No. 3, 2018, pp. 689–700.

[22] Al-Zubaidi, S., Stol, K., "Design Optimisation of Fully Actuated UAVs Using Hybrid Optimisation," *Journal of Intelligent & Robotic Systems,* Vol. 111, No. 1, 2025,

[23] Kim, T., Hong, S., "Control System Design and Experimental Validation of Hybrid Multicopter for Enhancement of Endurance," *Information,* Vol. 20, No. 12, 2017, pp. 8479–8486.

[24] Cyzba, R., Szafrański, G., Janik, M., "Development of Co-Axial Y6-Rotor UAV − Design, Mathematical Modeling, Rapid Prototyping and Experimental Validation," *International Conference on Unmanned Aircraft Systems (ICUAS),* June 2015*,* Denver, USA.

[25] Verbeke, J., Schutter, J.D., "Experimental maneuverability and agility quantification for rotary unmanned aerial vehicle," *International Journal of Micro Air Vehicles,* Vol. 10, No. 1, 2018, pp. 3–11.

[26] Al-Zubaidi, S., Stol, K., "Analysis of the Transient Response of UAV Rotors," *International Micro Aerial Vehicles Conference and Competition (IMAV).* September 2023,

A Framework for Indoor-Monocular Depth Estimation on UAVs Using a Thermal Camera

Carlos M. Perez-Rodriguez*[1], Cesar Martinez-Torres[1] , and Jose Martinez-Carranza[2]
[1] Universidad de las Americas Puebla, Puebla, Mexico
[2]Instituto Nacional de Astrofisica, Optica y Electronica, Puebla, Mexico

http://www.imavs.org/

ABSTRACT

Monocular depth estimation is essential for autonomous UAV navigation, especially under low-visibility conditions. While thermal cameras enable perception regardless of lighting, their use for geometric reasoning is challenging due to intensity variations driven by object temperature and emissivity.

We propose a novel framework that transforms thermal images into a temperature-invariant structural representation using line segments. This preserves scene geometry while discarding volatile thermal data. The structural image is then processed by the AnyDepth neural network to predict dense depth maps.

Experiments on the OdomBeyondVision dataset, featuring indoor UAV flights with thermal sensors, show that our method enables accurate depth estimation. This demonstrates the viability of decoupling geometry from thermal intensity, paving the way for robust UAV navigation in challenging scenarios such as search and rescue.

1 INTRODUCTION

Unmanned Aerial Vehicles (UAVs) are increasingly employed in critical missions such as search and rescue, environmental monitoring, and infrastructure inspection. A fundamental requirement for enabling autonomy in these contexts is the ability to perceive depth from visual inputs. Monocular depth estimation, which infers 3D structure from a single image, provides a lightweight solution that is particularly suitable for small UAVs where payload capacity is limited [1, 2].

While RGB-based depth estimation has seen significant progress with deep learning approaches [3, 4], thermal cameras remain underexplored for this task, despite their inherent advantages. Unlike RGB sensors, thermal imaging enables perception in total darkness, fog, smoke, or other visually degraded environments [5]. These capabilities make thermal cameras ideal for operations in hazardous or low-visibility scenarios. However, their use for geometric reasoning is challenging, as thermal intensity is governed by object temperature and emissivity rather than stable photometric features. This introduces issues such as thermal crossover and spatial inconsistency, which undermine the performance of conventional depth estimation models [6].

To address these limitations, prior research has attempted to fuse thermal and RGB data [7], or apply domain adaptation techniques to transfer knowledge from visible to thermal domains [5]. Nonetheless, these approaches often require additional sensors or paired data, increasing system complexity and cost. An alternative strategy is to extract meaningful geometric information directly from thermal images, thereby decoupling geometry from thermal intensity altogether.

In this work, we propose a novel indoor-monocular depth estimation framework tailored for thermal images. Our approach converts the input thermal image into a temperature-invariant structural representation by extracting line segments. This representation captures the essential scene geometry while discarding unstable thermal cues. The resulting structural image is then processed by the AnyDepth-v2 deep neural network [8, 9] to predict a dense depth map.

We evaluate our framework using the OdomBeyondVision dataset [10], which contains indoor UAV sequences recorded with thermal cameras. Experimental results show that our method enables accurate depth estimation from thermal input alone, without requiring additional sensors or modalities. This opens the door to robust indoor UAV navigation, particularly in scenarios where thermal sensors are already deployed (such as search and rescue, firefighting, and industrial inspection).

The remainder of this paper is organized as follows: Section 2 reviews related work on monocular depth estimation and thermal image processing. Section 3 details our proposed framework, including preprocessing, structural extraction, and the depth prediction pipeline. Section 4 presents our experimental setup, dataset, and evaluation results. Section 5 discusses the implications and limitations of our approach, and Section 6 concludes the paper with directions for future research.

2 RELATED WORK

2.1 Depth Estimation

The field of monocular depth estimation was revolutionized by deep learning, which has largely superseded classical geometric methods. The pioneering work of Eigen et al. [11] was the first to successfully demonstrate that a multi-scale Convolutional Neural Network (CNN) could regress a dense

*Email address(es): carlos.perezro@udlap.mx

depth map directly from a single RGB image in a supervised fashion. This established the foundational paradigm for end-to-end depth learning.

Shortly after, a second major paradigm emerged with the work of Garg et al. [12], which introduced self-supervised learning for this task. By using photometric consistency between stereo image pairs as a supervisory signal, they eliminated the need for expensive ground-truth data from active sensors like LiDAR. This breakthrough paved the way for modern, robust self-supervised methods. Among them, Monodepth2 [13] became a fundamental and widely-used baseline, thanks to its carefully designed loss function that handles occlusions and moving objects. Further refinements like ManyDepth [14] improved consistency by leveraging information from multiple frames during inference.

In the supervised domain, architectures like AdaBins [15] have achieved exceptional performance. Its main innovation is a novel method to discretize the depth range into adaptive bins, allowing the network to capture both the global scene layout and fine-grained local details with high fidelity.

A key challenge in the field is generalization to unseen environments and recovering true metric scale. Architectures like MiDaS [16] decisively addressed this by training on a mixture of diverse public datasets, achieving unprecedented zero-shot generalization capabilities. More recently, models such as AnyDepth [8] have pushed the state-of-the-art in generalization even further.

However, the success of all these state-of-the-art methods fundamentally relies on the photometric consistency inherent in RGB images, a property that is absent in data from thermal cameras, thus motivating our work.

3 PROPOSED FRAMEWORK

Our proposed framework is designed to compute a dense depth map from a single thermal image by transforming it into a robust, temperature-invariant structural representation. The methodology consists of a multi-stage process, including image pre-processing, line segment extraction, and entropy-based filtering, before feeding the result into a deep neural network. The overall pipeline is depicted in Figure 1.

3.1 Thermal Image Preprocessing and Normalization

The process begins with the raw thermal image in 16-bit format, T. To reduce sensor noise while preserving significant edges, a **bilateral filter** is applied. The value of a pixel $\mathbf{x}$ is updated according to:

$$T_f(\mathbf{x}) = \frac{1}{W_p} \sum_{\mathbf{y}\in\Omega(\mathbf{x})} T(\mathbf{y}) G_{\sigma_s}(\|\mathbf{x}-\mathbf{y}\|) G_{\sigma_r}(|T(\mathbf{x})-T(\mathbf{y})|) \tag{1}$$

where T_f is the filtered thermal image, $\Omega(\mathbf{x})$ is a neighborhood around $\mathbf{x}$, G_{σ_s} and G_{σ_r} are Gaussian kernels for the spatial and range (intensity) domains, respectively, and W_p is a normalization factor. Subsequently, to handle outliers, pixel values are clipped and then rescaled using min-max normalization into a 8-bit single-channel intensity map, T_{cn}.

3.2 Line Segment Extraction

The core of our framework is the conversion of the thermal image into a structural representation that is robust to thermal variations. For this, we extract line segments using the **Fast Line Detector (FLD)** algorithm [17]. However, the quality of the detected lines is highly dependent on the algorithm's internal parameters θ. For this work, we select a configuration of these parameters θ based on previous experiments, to obtain the line-segment representation of thermal image $L_{T(\theta)}$

This simplified approach was deliberately chosen to clearly demonstrate the viability of our core hypothesis without introducing the complexity of an advanced optimization routine. We note, however, that this selection process could be further enhanced by employing more sophisticated techniques, such as swarm intelligence algorithms [18, 19], which remains a promising direction for future work.

3.3 Entropy-Based Filtering for Noise Suppression

To suppress noisy line segments in flat regions, we use a filtering step based on local entropy. First, a **local entropy map**, E_Ω, is computed from T_f. For each pixel $\mathbf{x}$, its local entropy is calculated over a neighborhood window $\Omega_k(\mathbf{x})$ of size $k \times k$:

$$E_\Omega(\mathbf{x}) = -\sum_{j=0}^{255} p_j(\mathbf{x}) \log_2(p_j(\mathbf{x})) \tag{2}$$

where $p_j(\mathbf{x})$ is the probability of intensity level j within the window $\Omega_k(\mathbf{x})$. This map highlights true edges with high entropy values. This map is then thresholded to create a binary mask, M:

$$M_E(\mathbf{x}) = \begin{cases} 1 & \text{if } E_\Omega(\mathbf{x}) > \tau_e \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where τ_e is a predefined entropy threshold. The final, clean line segment image, L_f, is obtained by an element-wise product:

$$L_f = L_{T(\theta)} \odot M_E \tag{4}$$

where $\odot$ denotes the Hadamard product [20, 21].

3.4 Depth Estimation Network

The final, filtered line segment image, L_f, serves as the input to the depth estimation network. For this work, we only use the inference power of **AnyDepth-V2** to obtain the relative dense maps $\hat{D}$. The entire process can be seen as a function composition where the final depth map $\hat{D}$ is produced by the network $\mathcal{N}$ from the output of our pre-processing framework $\mathcal{F}$:

$$\hat{D} = \mathcal{N}(\mathcal{F}(T)) \quad \text{where} \quad \mathcal{F}(T) = L_f \tag{5}$$

http://www.imavs.org/

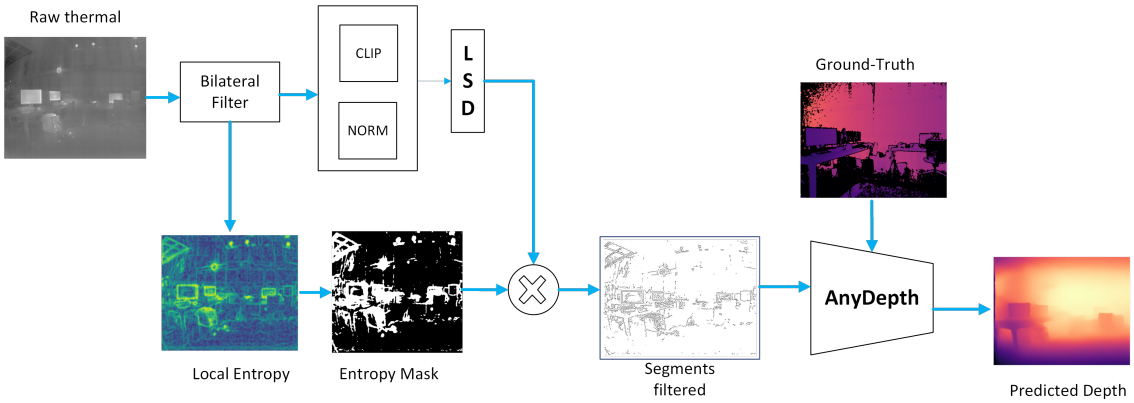


Figure 1: Pipeline of depth estimation from line-segment thermal representation.

4 Experimental and Results

This section presents the empirical validation of our framework. The main goal is to evaluate whether our proposed structural representation, derived from a thermal image, can serve as an effective input for a depth estimation network in the thermal domain, and to compare its performance against a standard RGB image input.

4.1 Experimental Setup

4.1.1 Dataset

For our experiments, we utilize the **OdomBeyondVision** dataset [22]. This indoor dataset provides sequences of RGB, depth and thermal images, captured from three platforms: Unmanned Ground Vehicle (UGV), Unmanned Aerial Vehicle (UAV) and Han-held. In this work, we use one UAV sequence, see details in Table 1. For the different frequency rates of devices, the number of images are different for the same sequence, so we made a synchronization of images using their time-stamps.

Sequence	Thermal	RGB	Depth
2022-01-20-00-14-36	18515	3176	6353

Table 1: UAV sequence details from OdomBeyondVision.

4.1.2 Implementation details

The pipeline was implemented in Python. The thermal images were first processed using a bilateral filter to reduce noise while preserving edges. We used the OpenCV library's implementation with a neighborhood diameter of `d=5`, and standard deviations of `sigmaSpace` (σ_s) = 100 and `sigmaColor` (σ_r) = 100.

Following this, the local entropy map was computed using the `entropy` function from Scikit-image, with a circular structuring element of radius 5 (`disk(5)`). An entropy mask was then generated by thresholding this map to filter out noisy line segments in low-texture regions. The threshold τ_e was selected automatically for every image using Otsu method [23].

The line segments were extracted using OpenCV's Fast Line Detector (FLD) implementation. The key parameters for the FLD were set as follows: `length_threshold=1`, `distance_threshold=0.1`, `canny_th1=70`, and `canny_th2=100`. Merging of lines was disabled to preserve raw structural details.

For the final depth estimation stage, we use the public implementation of the pre-trained **AnyDepth-V2 (Large)** model [8] in a zero-shot inference mode. Specifically, the model is configured with an output feature embedding size of $\mathbf{256}$ and utilizes multi-scale feature channels of $[\mathbf{256}, \mathbf{512}, \mathbf{1024}, \mathbf{1024}]$ in its decoder, as defined by the public repository. For inference, the structural input image $\mathcal{L}_f$ was resized to the model's default input resolution of $\mathbf{640} \times \mathbf{480}$ pixels. No fine-tuning or re-training was performed, as the purpose of the framework is to validate the effectiveness of the structural input alone.

4.1.3 Comparison Conditions:

To evaluate our method, we established two experimental conditions:

1. **Baseline (RGB):** The AnyDepth-V2 network receives the original RGB image from the dataset as input. This represents the expected performance of a standard vision system.

2. **Our Framework (Thermal-Structural):** The AnyDepth-V2 network receives the line segment representation generated from the corresponding thermal image, following the methodology described in Section 3.

http://www.imavs.org/

This direct comparison allows us to evaluate the viability of a structural representation derived from a thermal sensor for the task of depth estimation when only thermal data is available.

To quantitatively evaluate the performance of both conditions, we employ the six standard metrics for monocular depth estimation proposed in [11].

4.2 Results

The results of our evaluation on the test set are summarized in Table 2. The table compares the performance of our thermal-structural framework against the RGB baseline. As expected, the baseline using direct RGB input outperforms our framework across all evaluated metrics [11]. This is attributable to the fact that the AnyDepth-V2 network was trained extensively on images from the visible spectrum.

However, the most significant finding is not the performance gap, but that our thermal-structural framework produces a coherent and quantitatively reasonable depth map without any re-training or fine-tuning. The ability of a network trained exclusively on RGB data to interpret such a radically different input modality—a binary line segment image derived from a thermal sensor—and still infer the 3D structure of the scene, validates our core hypothesis: the proposed structural representation is a viable and meaningful input for depth estimation. This demonstrates the framework's potential to decouple geometric perception from thermal information.

A visual comparison of the results is presented in Figure 2. We show: (a) the original RGB image, (b) the thermal image, (c) our line segment representation, (d) the depth map predicted from the RGB input, (e) the depth map predicted by our framework, and (f) the ground truth.

5 DISCUSSION

The quantitative results presented in Table 2 show that the baseline, using direct RGB input, outperforms our framework. This outcome is expected, given that the AnyDepth-V2 network was pre-trained extensively on images from the visible spectrum. However, the most significant finding of this work is not the performance gap, but the demonstrated viability of our approach. The ability of a network trained exclusively on RGB data to interpret a radically different input modality—a binary line segment image derived from a thermal sensor—and still infer a coherent 3D structure validates our core hypothesis. This confirms that the proposed structural representation is a meaningful input for depth estimation, showcasing the potential to decouple geometric perception from volatile thermal information.

The structural representation is intentionally dense and subject to extreme parameter settings in the FLD algorithm to maximize the capture of geometric details. This configuration is a deliberate design choice: the exceptionally low *distance threshold* parameter (0.1) forces the detected elements to adhere to a strict model of rectilinearity. This crucial geometric constraint filters out the curvature and irregularity inherent in simple gradient-based thermal edges, ensuring that the input L_f is composed of stable structural information.

This framework opens possibilities for robust UAV navigation using a single, lightweight sensor in environments where RGB cameras would fail, such as darkness or smoke-filled rooms. For platforms where size, weight, and power (SWaP) are critical constraints, leveraging an existing thermal camera for both high-level tasks like human detection and low-level navigation is a significant advantage.

It is important to acknowledge the limitations of this study. A portion of the performance gap can be attributed to the fact that the RGB and thermal images in the OdomBeyondVision dataset are not perfectly spatially aligned, which introduces a systematic error in the evaluation. Furthermore, this validation was conducted in a dataset with adequate lighting conditions; the performance in the target scenario of complete darkness has not yet been empirically tested, nor has the generalization to unstructured outdoor aerial environments, which present challenges of greater range and variable scale.

6 CONCLUSION AND FUTURE WORK

In this paper, we presented a novel framework for monocular depth estimation on UAVs using a single thermal camera. Our core contribution is the transformation of thermal images into a temperature-invariant structural representation (line segments), addressing the lack of photometric consistency in thermal data. We successfully demonstrated that this structural representation serves as a viable direct input for the pre-trained AnyDepth-V2 network, validating the hypothesis that underlying geometric structure is a meaningful signal for depth perception. This opens a path for robust, lightweight, single-sensor navigation in challenging scenarios.

For future work, we identify three primary directions. First, the line segment extraction process can be enhanced; while the classic LSD algorithm is effective, we plan to explore modern deep learning-based methods for line detection, which could provide a more robust structural representation. Second, the most critical next step is to fine-tune or fully retrain a depth estimation network, such as AnyDepth, on our line segment representation. We hypothesize that this will significantly close the performance gap with the RGB baseline by allowing the network to specialize in this new modality. Finally, we will extend our validation to include datasets with challenging low-light and zero-light conditions to empirically demonstrate the framework's effectiveness in its primary target scenarios, such as search and rescue missions. This will include testing the framework's generalization capability to outdoor aerial environments.

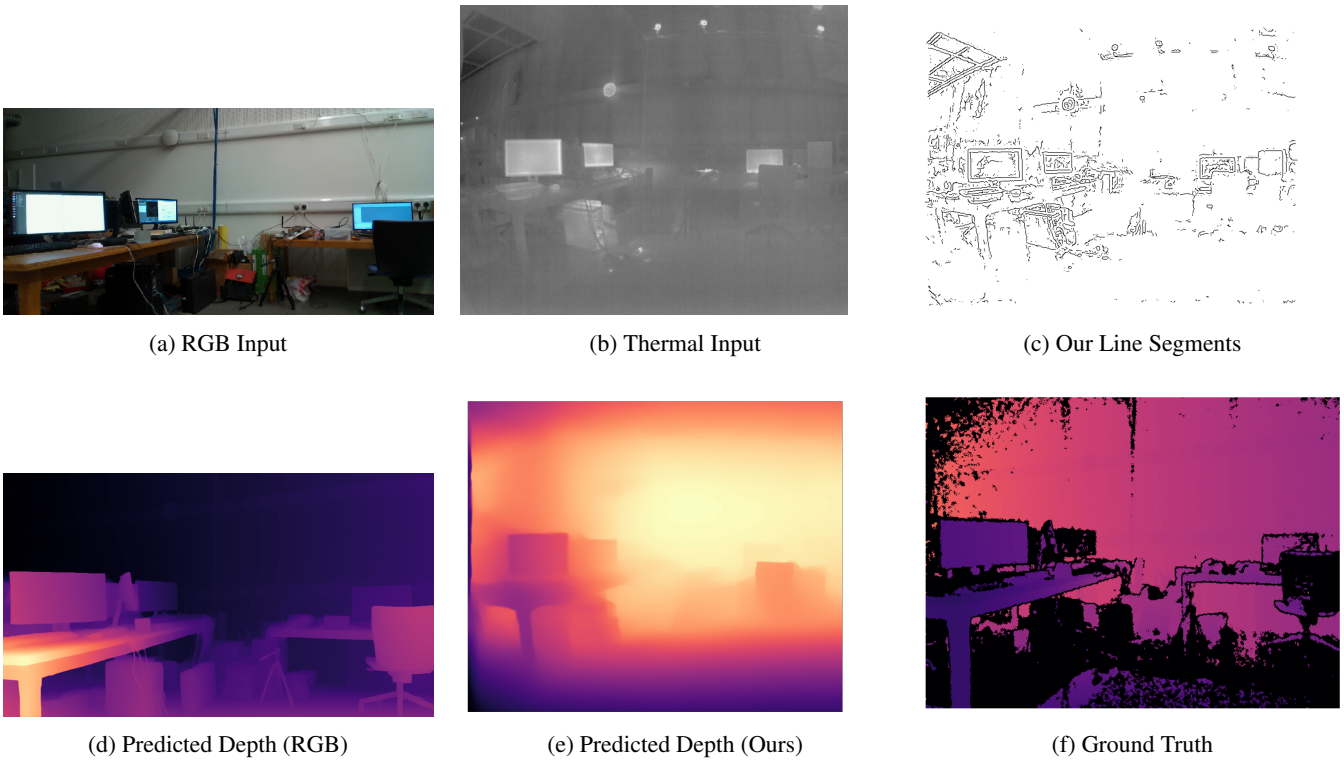
ACKNOWLEDGEMENTS

The authors thankfully acknowledge computer resources, technical advice and support provided by Laboratorio Nacional de Supercómputo del Sureste de México (LNS), a

http://www.imavs.org/

Method	Input Modality	REL ↓	RMSE ↓	log_{10} ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑
Baseline	RGB	0.3140	1.2442	0.3800	0.4955	0.7759	0.9015
Our Framework	**Lines (Thermal)**	0.4136	1.5156	0.4398	0.3981	0.6906	0.8635

Table 2: Quantitative comparison on the OdomBeyondVision dataset. Our framework uses a line segment representation from a thermal image as input to the AnyDepth-V2 network, while the baseline uses the corresponding RGB image.



(a) RGB Input (b) Thermal Input (c) Our Line Segments

(d) Predicted Depth (RGB) (e) Predicted Depth (Ours) (f) Ground Truth

Figure 2: Qualitative results. For a representative scene, we compare the depth map produced by the baseline using an RGB image against the depth map from our framework, which uses a structural representation derived from a thermal image. Our method successfully captures the main geometric structure of the scene.

http://www.imavs.org/

member of the CONAHCYT national laboratories, with project No. 202402046.

References

[1] Mingyang Lyu, Yibo Zhao, Chao Huang, and Hailong Huang. Unmanned Aerial Vehicles for Search and Rescue: A Survey. *Remote Sensing*, 15(13):3266, June 2023.

[2] Carlos Osorio Quero and Jose Martinez-Carranza. Unmanned aerial systems in search and rescue: A global perspective on current challenges and future applications. *International Journal of Disaster Risk Reduction*, 118:105199, February 2025.

[3] Zhiwei Huang, Mohammed A.H. Ali, Yusoff Nukman, Hai Lu Xu, Shikai Zhang, Hui Chen, and Mohammad Alkhedher. A systematic review of monocular depth estimation for autonomous driving: Methods and dataset benchmarking. *Results in Engineering*, 26:105359, June 2025.

[4] Jiuling Zhang. Survey on Monocular Metric Depth Estimation, 2025.

[5] Muhamad Risqi U. Saputra, Pedro P. B. De Gusmao, Chris Xiaoxuan Lu, Yasin Almalioglu, Stefano Rosa, Changhao Chen, Johan Wahlstrom, Wei Wang, Andrew Markham, and Niki Trigoni. DeepTIO: A Deep Thermal-Inertial Odometry With Visual Hallucination. *IEEE Robotics and Automation Letters*, 5(2):1672–1679, April 2020.

[6] Ukcheol Shin, Jinsun Park, and In So Kweon. Deep Depth Estimation from Thermal Image. In *2023 IEEE/CVF Conference on Computer Vision and Pat-*

tern Recognition (CVPR), pages 1043–1053, Vancouver, BC, Canada, June 2023. IEEE.

[7] Martin Brenner, Napoleon H. Reyes, Teo Susnjak, and Andre L. C. Barczak. RGB-D and Thermal Sensor Fusion: A Systematic Literature Review. *IEEE Access*, 11:82410–82442, 2023.

[8] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth Anything V2, 2024.

[9] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10371–10381, Seattle, WA, USA, June 2024. IEEE.

[10] Peize Li, Kaiwen Cai, Muhamad Risqi U. Saputra, Zhuangzhuang Dai, and Chris Xiaoxuan Lu. OdomBeyondVision: An Indoor Multi-modal Multi-platform Odometry Dataset Beyond the Visible Spectrum. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3845–3850, Kyoto, Japan, October 2022. IEEE.

[11] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2366–2374, Cambridge, MA, USA, 2014. MIT Press. event-place: Montreal, Canada.

[12] Ravi Garg, Vijay Kumar B.G., Gustavo Carneiro, and Ian Reid. Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, volume 9912, pages 740–756. Springer International Publishing, Cham, 2016.

[13] Clement Godard, Oisin Mac Aodha, Michael Firman, and Gabriel Brostow. Digging Into Self-Supervised Monocular Depth Estimation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3827–3837, Seoul, Korea (South), October 2019. IEEE.

[14] Jamie Watson, Oisin Mac Aodha, Victor Prisacariu, Gabriel Brostow, and Michael Firman. The Temporal Opportunist: Self-Supervised Multi-Frame Monocular Depth. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1164–1174, Nashville, TN, USA, June 2021. IEEE.

[15] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. AdaBins: Depth Estimation Using Adaptive Bins. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4008–4017, Nashville, TN, USA, June 2021. IEEE.

[16] Rene Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision Transformers for Dense Prediction. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12159–12168, Montreal, QC, Canada, October 2021. IEEE.

[17] Rafael Grompone Von Gioi, Jérémie Jakubowicz, Jean-Michel Morel, and Gregory Randall. LSD: a Line Segment Detector. *Image Processing On Line*, 2:35–55, March 2012.

[18] Tareq M. Shami, Ayman A. El-Saleh, Mohammed Alswaitti, Qasem Al-Tashi, Mhd Amen Summakieh, and Seyedali Mirjalili. Particle Swarm Optimization: A Comprehensive Survey. *IEEE Access*, 10:10031–10061, 2022.

[19] Thounaojam Chinglemba, Soujanyo Biswas, Debashish Malakar, Vivek Meena, Debojyoti Sarkar, and Anupam Biswas. Introductory Review of Swarm Intelligence Techniques. In Anupam Biswas, Can B. Kalayci, and Seyedali Mirjalili, editors, *Advances in Swarm Intelligence*, volume 1054, pages 15–35. Springer International Publishing, Cham, 2023.

[20] Roger A. Horn and Charles R. Johnson. *Matrix analysis*. Cambridge University Press, New York, NY, second edition, corrected reprint edition, 2017.

[21] Grigorios G. Chrysos, Yongtao Wu, Razvan Pascanu, Philip H.S. Torr, and Volkan Cevher. Hadamard Product in Deep Learning: Introduction, Advances and Challenges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(8):6531–6549, August 2025.

[22] Peize Li, Kaiwen Cai, Muhamad Risqi U. Saputra, Zhuangzhuang Dai, Chris Xiaoxuan Lu, Andrew Markham, and Niki Trigoni. OdomBeyondVision: An Indoor Multi-modal Multi-platform Odometry Dataset Beyond the Visible Spectrum, 2022.

[23] Nobuyuki Otsu. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, January 1979.

http://www.imavs.org/

Energy-Aware Hybrid Event-Triggered Control for Micro-Drones

Jesus O. Rodriguez-Ramirez[1]0009-0007-2817-1329, Julio S. De La Trinidad-Rendon[1]0000-0003-3900-8563 , and Jose Martinez-Carranza[2]0000-0002-8914-1904 , and Hugo G. Gonzalez-Hernandez[1]0000-0001-6495-9980*
1 Tecnologico de Monterrey, School of Engineering and Sciences, Puebla, Mexico
2 Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), Puebla, Mexico

http://www.imavs.org/

ABSTRACT

This paper addresses the important balance between control performance and energy use in networked micro-drones. We propose and test a family of hybrid event-triggered control (HETC) methods that are designed to be both dependable and efficient. The core of our strategy is the combination of a fixed safety timer, which ensures system safety, with an adjustable threshold that responds to the drone's state to balance communication, performance, and energy. We look at several practical ways to adjust the threshold, including methods based on integral, derivative, and state-feedback approaches. The effectiveness of these strategies is tested using detailed simulations of a 6-DOF quadrotor. The validation uses dour different flying paths which include active tracking in straight and curved trajectories as well as stationary hovering, to test the controller in various conditions. The results show meaningful reductions not only in communication frequency but also in a measure of total energy use. This study shows our practical HETC approach is a promising solution for devices with limited resources and provides a solid starting point for future research on more advanced triggers and testing with real-world network problems.

1 INTRODUCTION

The increasing autonomy of Micro Aerial Vehicles (MAVs) presents a modern control challenge: greater autonomy and performance require more advanced onboard computing and frequent communication, yet the physical limits of these devices, especially battery life, require careful use of resources [1]. This problem is most significant in Networked Control Systems (NCS), where MAVs must constantly exchange information with ground stations or other agents, making the energy used for communication a major limit for mission duration.

The traditional time-triggered control method, which uses updates at regular, frequent intervals, is a major source of this inefficiency. By sending control data at a fixed rate based on a worst-case scenario, these systems waste energy sending unneeded data during normal flight. Event-Triggered Control (ETC) solves this by changing the control approach from updating at set times to updating only when necessary [2, 3]. This non-periodic approach has been shown to greatly reduce network use, a topic explored in detailed reviews on ETC for nonlinear systems [4, 5]. Many adaptive/learning or CLF-based triggers require evaluating Lyapunov/ISS bounds (and their Lie derivatives) online and, in some cases, solving small optimization problems (e.g., QP/SDP) or running neural inference at control rates. These steps increase CPU load, memory footprint, and code size, which can exceed MAV-class microcontrollers. See emulation/self-triggered and PETC analyses in [6, 7, 8].

Research has shifted from fixed triggers to flexible and adaptive methods that are more efficient, including those based on fuzzy neural networks [9] and reinforcement learning [10]. For Unmanned Aerial Vehicles (UAVs) specifically, ETC has been applied to periodic communication schemes [11, 12], swarm control [13], and security [14]. A key challenge is making sure it performs well even with constant outside disturbances [15, 16] and modeling errors, which has led to more research on dependable and adaptive ETC for autonomous vehicles in general [17, 18].

Despite this progress, two key problems are often overlooked. First, many studies only focus on sending fewer data packets, ignoring the total energy used, which includes the energy the motors use to fix trajectory-tracking errors. A trigger that sends updates too slowly might save on communication, but lead to poor trajectory-tracking that requires powerful, energy-intensive corrections. Second, many of the best-performing trigger rules from theory can be too complex to run, making them hard to use on MAVs with limited computing power. By contrast, our rules use only vector norms, constant-gain arithmetic, and (optionally) first-order filters—no optimizers, factorizations, or large matrices—so the per-step cost is $O(1)$ floating-point operations, appropriate for Cortex-M–class MCUs [8].

This paper explores a practical alternative: a set of simple heuristic adaptive triggers, when used in a dependable hybrid system, can save a large amount of energy without the complexity of other methods. We propose a Hybrid Event Triggered Control (HETC) framework that combines an adaptive

*Email address(es): hgonz@tec.mx

http://www.imavs.org/

event-trigger with a fixed safety timer ($T_{\max}$) to guarantee stability. Our main goal is to measure the balance between trajectory-tracking accuracy, communication frequency, and, most importantly, total energy use.

1.1 Quadrotor Dynamic Model

We model the quadrotor as a 6-DOF rigid body. Let $\{W\}$ be the fixed world frame and $\{B\}$ be the frame attached to the drone's body. The state is $(\boldsymbol{p}, \boldsymbol{v}, \boldsymbol{R}, \boldsymbol{\Omega}) \in \mathrm{SE}(3)(3) \times \mathbb{R}^3$, where $\boldsymbol{p}, \boldsymbol{v} \in \mathbb{R}^3$ are the position and velocity in $\{W\}$, $\boldsymbol{R} \in \mathrm{SO}(3)$ is the rotation matrix from $\{B\}$ to $\{W\}$, and $\boldsymbol{\Omega} \in \mathbb{R}^3$ is the angular velocity in $\{B\}$. The dynamics are given by [19]:

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \tag{1}$$

$$m\dot{\boldsymbol{v}} = mg\boldsymbol{e}_3 - \boldsymbol{R}T\boldsymbol{e}_3 + \boldsymbol{d}_f(t) \tag{2}$$

$$\dot{\boldsymbol{R}} = \boldsymbol{R}\hat{\boldsymbol{\Omega}} \tag{3}$$

$$\boldsymbol{J}\dot{\boldsymbol{\Omega}} = -\boldsymbol{\Omega} \times (\boldsymbol{J}\boldsymbol{\Omega}) + \boldsymbol{\tau} + \boldsymbol{d}_\tau(t) \tag{4}$$

where m is the mass, $\boldsymbol{J}$ is the inertia matrix, g is gravity, $\boldsymbol{e}_3 = [0, 0, 1]^T$, T is the total thrust, and $\boldsymbol{\tau}$ is the torque vector. The terms $\boldsymbol{d}_f(t)$ and $\boldsymbol{d}_\tau(t)$ are bounded external disturbances. The hat map $\hat{\cdot}$ is the skew-symmetric matrix representation.

1.2 Baseline Control Architecture and Implementation

Our method builds on a standard nested-loop controller:

- **Inner Attitude Loop:** A high-rate controller (e.g., 200 Hz or higher) regulates the drone's rotation by computing the motor torques $\boldsymbol{\tau}$ required to track a desired orientation $\boldsymbol{R}_d(t)$.
- **Outer Position Loop:** A continuous-time nonlinear controller (e.g., geometric on $\mathrm{SE}(3)(3)$) tracks the reference path $\boldsymbol{p}_d(t)$ and its derivatives. Its output is the nominal control signal $\boldsymbol{u}(t) = [T(t), \boldsymbol{R}_d(t)]^T$.

In practice, $\boldsymbol{u}(t)$ cannot be transmitted continuously due to communication and energy constraints. Instead, it is sampled at discrete times $\{t_k\}_{k\in\mathbb{N}_0}$, which the proposed HETC framework determines.

Between update instants, the motors apply the most recently transmitted command using a Zero-Order Hold (ZOH) mechanism. The executed control signal is therefore

$$\boldsymbol{u}_{drone}(t) = \boldsymbol{u}(t_k), \quad \forall t \in [t_k, t_{k+1}), \tag{5}$$

i.e., constant between successive triggers. This implementation introduces a discrepancy between the nominal signal $\boldsymbol{u}(t)$ and the applied signal $\boldsymbol{u}_{drone}(t)$, defined as the measurement error $\boldsymbol{e}_m(t) = \boldsymbol{u}(t) - \boldsymbol{u}_{drone}(t)$. The event-triggering mechanism monitors and regulates this error.

1.3 Total Energy Use Model

System efficiency is evaluated with an energy model that accounts for both communication and actuation costs:

$$E = \int_0^{t_f} P_{base}\, dt + N_{pkts}\, E_{tx} + \int_0^{t_f} P_{act}(\boldsymbol{u}_{drone}(t))\, dt, \tag{6}$$

where P_{base} is the constant baseline consumption of onboard electronics, N_{pkts} the number of transmitted packets, and E_{tx} the energy per packet. Since P_{base} is independent of the control scheme, comparisons focus on:

- **Communication Energy (E_{comm}):** $N_{pkts}\, E_{tx}$, reduced by event-triggering strategies.
- **Actuation Energy (E_{act}):** The integral of motor power, commonly approximated as $P_{act} \propto T(t)^2$.

This formulation highlights the trade-off: frequent updates improve tracking but increase communication cost, whereas sparse updates reduce communication but may require larger corrective maneuvers and thus higher motor energy. Effective triggers balance these contributions to minimize $E_{comm} + E_{act}$.

The energy model in Eq. 6, which separates baseline, communication, and actuation terms, follows standard UAV power formulations and ETC studies that include communication cost [1, 11, 12].

2 PROPOSED HYBRID ADAPTIVE CONTROL FRAMEWORK

The framework is intended to be reliable, efficient, and suitable for resource-constrained platforms.

If the baseline continuous controller guarantees closed-loop stability under sample-and-hold with period $h \le T_{\max}$ (standard emulation condition), then enforcing the time guard preserves this property. From sampled-data and PETC theory, the resulting hybrid loop is input-to-state stable (ISS) with respect to disturbances and measurement error. The event condition further improves accuracy between mandatory watchdog updates [8, 7, 6].

2.1 Hybrid Event- and Time-Triggering Law

The triggering mechanism combines an event condition with a periodic safeguard. Let t_k denote the last update time and $\boldsymbol{e}_m(t) = \boldsymbol{u}(t) - \boldsymbol{u}(t_k)$ the measurement error. The next update time t_{k+1} is defined as

$$t_{k+1} = \inf\{t > t_k \mid \underbrace{\|\boldsymbol{e}_m(t)\| \ge \sigma(t)}_{\text{Event Condition}} \vee \underbrace{t - t_k \ge T_{\max}}_{\text{Time Guard}}\}. \tag{7}$$

The event condition enforces updates when the control error exceeds the adaptive threshold $\sigma(t)$, while the time guard $T_{\max}$ prevents excessive inter-update intervals. Together, these conditions guarantee stability and avoid unnecessary communication.

http://www.imavs.org/

2.2 Adaptive Threshold Structure

The threshold $\sigma(t)$ is defined using the error vector $\boldsymbol{x}_e = [\boldsymbol{e}_p^T, \boldsymbol{e}_v^T]^T$:

$$\sigma(t) = \underbrace{\beta(t)}_{\text{Adaptive Offset}} + \underbrace{\alpha \left\|\boldsymbol{e}_p(t)\right\| + \gamma \left\|\boldsymbol{e}_v(t)\right\|}_{\text{Error-Dependent Scaling}}, \tag{8}$$

where α and γ are fixed gains, and $\beta(t)$ is a time-varying offset. The offset term allows the trigger sensitivity to be adjusted online according to mission demands or system behavior.

The gains α, γ, and the adaptation parameters were chosen to balance trigger sensitivity and stability. Values were tuned to maintain RMS tracking error below $0.1\,\text{m}$ while avoiding excessive updates and preserving stability, following the typical trade-offs reported in event-triggered UAV literature.

To implement this mechanism, we introduce three alternative adaptation laws for $\beta(t)$, each emphasizing a different objective:

2.3 Adaptation Rule 1: Integral Control for Rate Regulation

Objective Regulate the communication rate by driving the average number of updates, N_{avg}, toward a target N^*.

Rule An integral controller with leakage:

$$\dot{\beta}(t) = -k_I\big(N_{\text{avg}}(t) - N^*\big) - \lambda\big(\beta(t) - \beta_0\big), \tag{9}$$

where $\beta(t)$ is the adaptive offset. Increasing β raises the trigger threshold, reducing update frequency; decreasing β lowers the threshold, increasing updates. The leakage term, governed by λ, prevents windup by returning β toward a nominal value β_0 when the rate stabilizes.

Implementation Considerations Performance depends on the averaging window used for N_{avg}: shorter windows improve responsiveness but amplify noise, whereas longer windows provide smoother estimates at the cost of slower adaptation. Gains k_I and λ must be tuned to achieve stability without oscillation.

2.4 Adaptation Rule 2: Derivative Feedback

Objective To adjust the trigger based on the rate of change of the position error, increasing sensitivity when the error grows and reducing it when the error decreases.

Rule

$$\beta(t) = \beta_0 - k_D \frac{d}{dt} \left\|\boldsymbol{e}_p(t)\right\|, \tag{10}$$

where a positive derivative decreases the offset β, making an update more likely, and a negative derivative increases it, reducing update frequency. This predictive adjustment helps limit maximum error growth during disturbances and can lower the total actuation energy required for correction.

Implementation Considerations Direct differentiation is noise-sensitive. A filtered derivative estimate (e.g., via a first-order high-pass filter or observer) is required. The filter cutoff frequency must be chosen carefully to balance responsiveness and robustness.

2.5 Adaptation Rule 3: State-Feedback for Context-Awareness

Objective Define a rule that adapts to the drone's operating condition (e.g., transit versus hover) without requiring memory of past data or derivative estimates.

Rule

$$\beta(t) = \beta_0 + k_p \left\|\boldsymbol{p}(t)\right\| + k_v \left\|\boldsymbol{v}(t)\right\|, \tag{11}$$

where $\boldsymbol{p}(t)$ and $\boldsymbol{v}(t)$ are the position and velocity. When the drone is moving quickly or operating farther from the target, the threshold increases, reducing communication demand. During hovering or precise maneuvers, the threshold decreases, prioritizing accuracy.

Implementation Considerations The parameters k_p and k_v must be scaled to the mission's workspace and expected speeds. This rule is less adaptable to changing tasks than the other laws but remains straightforward and reliable.

3 SIMULATION STUDY AND DISCUSSION OF RESULTS

3.1 Simulation Environment

The proposed HETC framework was implemented in MATLAB/Simulink using a 6-DOF quadrotor model with parameters corresponding to the Crazyflie 2.1 micro-drone ($m = 0.031$ kg).

As a baseline, we employed a geometric tracking controller on $\text{SE}(3)(3)$ [19], which directly regulates both position and orientation.

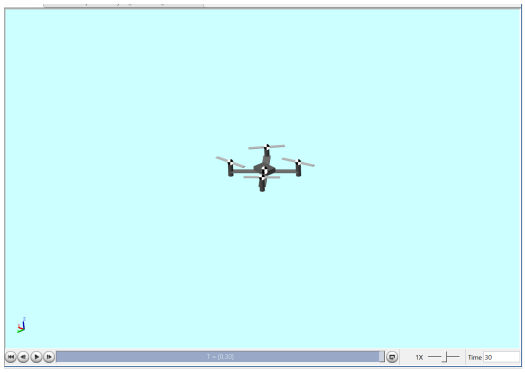


Figure 1: Simulation environment used for the evaluation of the proposed hybrid event-triggered control.

Figure 1 shows the simulated quadrotor environment used to test the proposed control scheme.

http://www.imavs.org/

Performance and energy efficiency were evaluated on a set of 30-seconds trajectory experiments designed to combine periods of active motion and near-stationary hovering. This provided a test of controller behavior under varied conditions. Four representative trajectories were considered:

- **Circle:** A horizontal circular path of 1m radius at constant altitude.
- **Lemniscate:** A figure-eight path designed to test adaptability.
- **Square:** A stepwise path with sharp corners.
- **L path:** A piecewise-linear "L-shaped" trajectory with alternating segments.

Together, these trajectories provide a controlled testbed for analyzing how the hybrid triggering mechanism balances communication savings and actuation energy across diverse operating conditions. Reported results for UAV/VTOL systems show 40–80% packet-rate reductions with less than 10–15% degradation in tracking accuracy under ETC and PETC schemes [3, 4, 5, 11, 12, 13].

3.2 *Discussion of Results*

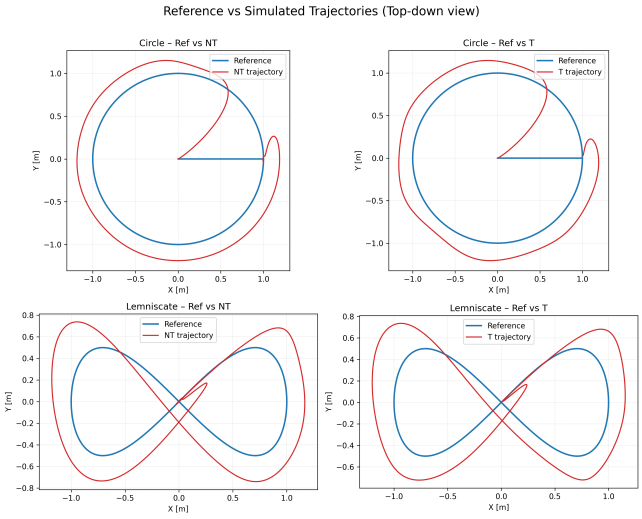


Figure 2: Top-down view of the reference and simulated trajectories for circular and lemniscate paths.

Figure 2 compares the reference and simulated trajectories for the circular and lemniscate paths under non-triggered and hybrid event-triggered control. The NT trajectories correspond to the non-triggered (periodic) controller, while T trajectories correspond to the hybrid event-triggered implementation.

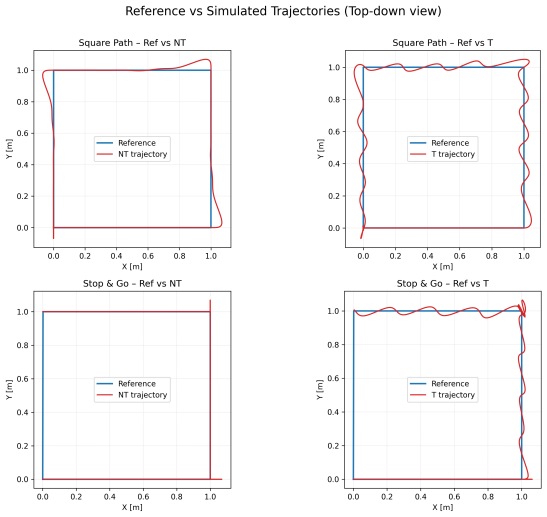


Figure 3: Comparison of reference and simulated trajectories for the square and stop-and-go paths.

Figure 3 illustrates trajectory tracking for square and stop-and-go paths, showing the effect of the hybrid trigger compared to the fixed-rate baseline. The non-triggered (NT) case uses fixed-rate updates, while the triggered (T) case applies the proposed hybrid event-triggered policy.

Straight-line segments exhibit small oscillations because low curvature reduces trigger activity, causing delayed corrections, whereas curved paths naturally maintain frequent updates and smoother tracking, the oscillations also contribute greatly to an increase on actuation energy as shown on table 1.

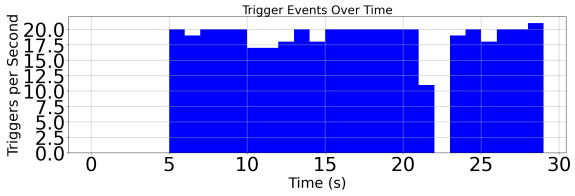


Figure 4: Visualization of the HETC trigger signal over the a multi-phase trajectory to summarize different trials. The histogram shows the number of trigger events on that second

The simulation results confirm that the proposed HETC framework reduces resource consumption while maintaining tracking performance. The observed behavior is consistent across the evaluated trajectories and ablation studies.

3.2.1 Analysis of Triggering Behavior

Figure 4 illustrates the distribution of trigger events over time. The trigger activity closely follows the multi-phase trajectory:

- **Dynamic Tracking:** Frequent updates are generated to maintain path accuracy, resembling periodic control. Small gaps indicate instances where communication is avoided without performance loss.

- **Hovering:** With the drone stabilized, the tracking error remains negligible and the trigger rarely activates, leading to substantial communication savings relative to periodic control.

- **Reorientation and Settling:** The yaw maneuver induces a transient error, producing a short burst of updates. After the maneuver, the trigger rate decreases again as the system stabilizes.

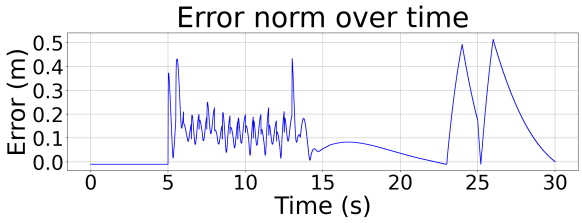


Figure 5: Evolution of the position tracking error norm, $||e_p(t)||$ (in meters) vs Time in seconds

The evolution of the position error norm corresponds closely to the trigger activity shown in Fig. 4:

- **Dynamic Tracking:** During trajectory following, the error remains small (typically below 0.3 m) with an oscillatory pattern characteristic of continuous corrective actions. The high trigger rate is consistent with the need for frequent adjustments.

- **Hovering:** Once stabilized in hover, the tracking error quickly decreases to near zero, confirming that extended periods without triggers occur because the control objective is maintained.

- **Yaw Reorientation:** A reorientation maneuver produces temporary error spikes of up to approximately 0.55 m, after which the controller restores accuracy, demonstrating robustness to internal disturbances.

- **Final Settling:** Following the maneuver, the error again converges to near zero, and the system resumes stable, low-energy operation.

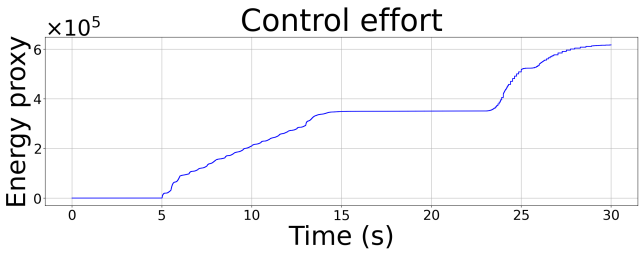


Figure 6: Cumulative actuation energy (E_{act}) vs Time in seconds, proxied by the integral of the squared output of the controllers with no HETC.

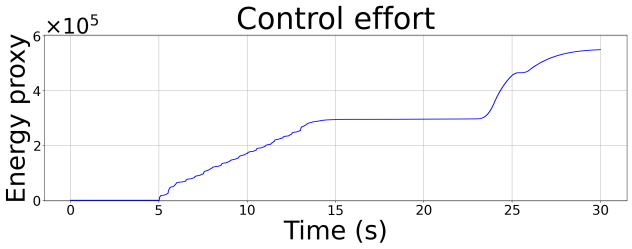


Figure 7: Cumulative actuation energy (E_{act}) vs Time in seconds, proxied by the integral of the squared output of the controllers with HETC.

The best way to measure the system's success is its impact on total energy use, as modeled in Eq. (6). While the analysis of the trigger signal in Fig. 4 confirmed a large reduction in communication energy (E_{comm}), a detailed look at the motor energy (E_{act}) shows an even better result.

Figures 6 and 7 present a comparison of the cumulative motor energy for the baseline periodic controller, The motor energy is estimated by calculating the integral of the squared thrust command over the simulation time, the first one being the system without our event triggering logic.

Looking closely at the plots shows two key things:

- **Similar Power Profiles:** Both the periodic and HETC systems show similar power usage patterns that match the mission phases. As expected, energy use increases rapidly during active flight and is almost flat when hovering.

- **Reduced Total Actuation Energy:** But there is a big difference in the total energy used. The periodic controller's motor energy continually climbs, reaching a final value of about 3.5 x10^5 units. In contrast, the HETC system's total motor energy is lower, settling around 3.0 x10^5 units. This is a reduction of about 14% in the energy used by the motors.

http://www.imavs.org/

Table 1: ΔPkts [%] = packet reduction, E-cons = Energy consumed by both communication and actuation

Trajectory	ΔPkts [%]	E-cons	RMS error [m]
Circle (R=1 m)	28	↓	0.07
Lemniscate	24	↓	0.06
Square (step corners)	39	↑↑	0.018
L path (piecewise-linear)	41	↑↑	0.022

Table 1 summarizes the results per trajectory obtained with the proposed HETC compared to the periodic baseline. For smooth trajectories such as the circle and lemniscate, communication demand is reduced by about 25–30% with negligible loss of tracking accuracy (RMS error below 0.1 m) and even slight energy savings, since fewer corrective maneuvers are required. In contrast, trajectories with sharp corners or long straight segments (square and L-path) achieve higher packet savings (around 40%), but this comes at the cost of increased actuation effort. The oscillatory corrections required in these cases lead to additional motor energy consumption (indicated qualitatively in the table), even though the RMS error remains low. These results highlight the trade-off: smoother paths yield balanced savings across communication and energy, while step-like trajectories shift the burden toward actuation, reducing the overall energy benefit.

4 CONCLUSION AND FUTURE WORK

This paper presented a hybrid control framework that combines event- and time-based triggers to address energy efficiency in MAVs. By integrating a safety timer with simple adaptive trigger rules, the method balances control performance with reduced communication and actuation energy.

The simulation study, based on a total energy model, shows that these adaptive strategies lower both communication frequency and overall energy consumption compared to periodic and fixed-threshold approaches. All proposed rules are applied within the same framework, ensuring practicality for MAV-class hardware.

Future work will focus on extending the evaluation beyond ideal simulations to include more realistic network conditions with delays and packet losses, as well as hardware-in-the-loop experiments. Another direction is the development of triggers derived from control-theoretic formulations, such as those based on Control Lyapunov Functions (CLFs), with the goal of assessing whether their theoretical performance gains justify the additional computational complexity on embedded platforms.

Moreover an important limitation observed in the present study is that the triggering mechanism and control design do not explicitly account for the inertia of the vehicle. As a result, aggressive maneuvers and step-like trajectory segments often introduce large overshoots and oscillations, which in turn increase actuation energy consumption. While the proposed adaptive thresholds successfully regulate communication effort, they are less effective in anticipating the high corrective demands associated with inertial dynamics. Future work will focus on incorporating inertia-aware terms into the triggering condition, such as acceleration, jerk, or command-rate penalties, to prevent delayed updates during rapid maneuvers. By making the event-trigger more sensitive to inertial effects, it is expected that both overshoot and oscillatory energy losses can be reduced, improving overall efficiency without compromising tracking performance.

REFERENCES

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

[2] Paulo Tabuada. Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Transactions on Automatic Control*, 52(9):1680–1685, 2007.

[3] WPMH Heemels, KH Johansson, and P Tabuada. An introduction to event-triggered and self-triggered control. *Proceedings of the 51st IEEE Conference on Decision and Control*, pages 3270–3285, 2012.

[4] Rida Yaqoob and et al. A recent survey of event-triggered control of nonlinear systems. *arXiv preprint arXiv:2202.01684*, 2021.

[5] Derui Ding, Qing-Long Han, and et al. Dynamic event-triggered control and estimation: A survey. *International Journal of Systems Science*, 52(13):2353–2376, 2021.

[6] Alvaro Anta and Paulo Tabuada. To sample or not to sample: Self-triggered control for nonlinear systems. *Automatica*, 46(5):949–958, 2010.

[7] Arjan Postoyan, Paulo Tabuada, Dragan Nešić, and Alvaro Anta. A framework for event-triggered and self-triggered control. *IEEE Transactions on Automatic Control*, 60(10):2661–2675, 2015.

[8] W. P. M. H. Heemels, M. C. F. Donkers, and A. R. Teel. Periodic event-triggered control for linear systems. *IEEE Transactions on Automatic Control*, 58(4):847–861, 2013.

[9] Kangkang Sun, Jianbin Qiu, Hamid Reza Karimi, and Yili Fu. Event-triggered robust fuzzy adaptive finite-time control of nonlinear systems with prescribed performance. *IEEE Transactions on Fuzzy Systems*, 29(6):1460–1471, 2021.

[10] Umair Siddique, Hassan Khalil, and Xiang Xu. Adaptive event-triggered reinforcement learning control for complex nonlinear systems. *arXiv preprint arXiv:2403.12069*, 2024.

http://www.imavs.org/

[11] Ángel Cuenca, Oscar Reinoso, and Pablo Gil. Periodic event-triggered sampling and dual-rate control for a wireless ncs with applications to uavs. *IEEE Transactions on Industrial Electronics*, 65(8):6510–6520, 2018.

[12] Tong WU, Jie WANG, and Bailing TIAN. Periodic event-triggered formation control for multi-uav systems with collision avoidance. *Chinese Journal of Aeronautics*, 35(8):193–203, 2022.

[13] Xiuxia Yang, Hao Yu, Yi Zhang, and Wenqiang Yao. Distributed event-triggered control for uav swarm target fencing with network connectivity preservation and collision avoidance. *Defence Technology*, 2025.

[14] Shijie Han, Chen Wang, and Xiaoyu Liu. Event-triggered attitude controller design for uavs under cyber-attacks. In *IEEE International Symposium on Industrial Electronics (ISIE)*, pages 1099–1104. IEEE, 2024.

[15] Peng Zhao, Yuan Liu, and Han Xu. Event-triggered trajectory tracking control for quadrotor uavs subject to external disturbances. *International Journal of Robust and Nonlinear Control*, 2025. Early Access.

[16] Liang Kong, Jun Wu, and Tianyang Zhang. Robust event-triggered tracking control for a uav with non-vanishing disturbances. *Automatica*, 159:111224, 2024.

[17] Hongguang Meng, Derui Ding, Hao Su, and Li Wang. Event-triggered robust output regulation for nonlinear systems: A time regularization approach. *International Journal of Robust and Nonlinear Control*, 32(17):7403–7421, 2022.

[18] Mehmet Arsal and Tansel Yucelen. Adaptive event-triggered consensus control for auvs with output feedback and energy efficiency. *Journal of Marine Science and Engineering*, 12(4):511, 2024.

[19] Taeyoung Lee, Melvin Leok, and N. Harris McClamroch. Geometric tracking control of a quadrotor UAV on SE(3). In *49th IEEE Conference on Decision and Control (CDC)*, pages 5420–5425, Atlanta, GA, USA, 2010. IEEE.

http://www.imavs.org/

Payload State Estimation for Cooperative Manipulation Using Multiple UAVs

Yuki Ohira, Abner Asignacion Jr., and Satoshi Suzuki*
Chiba University, Japan

http://www.imavs.org/

Abstract

This paper proposes a method for estimating the state of a payload in a cooperative aerial manipulation system using multiple UAVs connected via tethers. The key feature of the proposed approach is that it estimates the tether angles solely from inertial and position data obtained on the UAV side, eliminating the need for any sensors on the payload. These estimated angles are then utilized, in conjunction with a physical model, to infer the payload's position and orientation. The estimation framework is implemented using an Unscented Kalman Filter (UKF) and applied to a system comprising two UAVs and four tethers. The performance of the method is evaluated through both simulations and real-world experiments. The results demonstrate that the proposed approach achieves accurate tracking of both the tether angles and the payload state, even in the presence of sensor noise and modeling inaccuracies, thus validating its effectiveness.

1 Introduction

In recent years, there has been a growing global interest in small unmanned aerial vehicles (UAVs). Their applications extend beyond conventional fields such as aerial photography, disaster surveys, and infrastructure inspection, and are now being explored for more advanced and complex tasks [1]. Specifically, these include aerial manipulation using UAVs equipped with manipulators [2], and payload transportation in the field of logistics [3]. UAVs are expected to bring numerous benefits to society, including reduced costs and labor requirements, as well as safe access to environments that are difficult or dangerous for humans to enter.

However, most current UAVs are designed for single-unit operation and are therefore limited in the size and weight of payloads they can carry. To address these limitations, cooperative manipulation systems, in which multiple UAVs work together to transport a single payload, have garnered significant attention [4]. Such systems enable the handling of large and heavy payloads that are unmanageable by a single UAV, thereby greatly expanding the potential applications of UAVs.

In cooperative manipulation systems, a widely adopted configuration is to connect each UAV to the payload using cables [5]. While this approach allows for operational flexibility, it also introduces challenges due to the underactuated and deformable nature of the system. These include the risk of inter-UAV collisions, payload oscillations, and instability during flight. Therefore, ensuring the safe and stable operation of such systems requires accurate estimation of the full system state, including the positions and orientations of the UAVs and the payload, as well as cable tensions and angles [6, 7, 8].

Traditionally, estimating the system or payload state has relied on attaching various sensors to the payload. However, these methods face several practical challenges, including the need for dedicated power sources, communication infrastructure, increased cost and weight, and physical constraints on sensor installation. For instance, GPS and IMUs have been used to track the payload's motion [8], while force/torque sensors [9] and vision-based systems [10, 11] have enabled more precise measurements. Even when lightweight sensors are used, the added mass can be non-negligible in small-scale aerial systems with limited payload capacity. In addition, securing power and ensuring reliable data transmission for such sensors impose further constraints that reduce the practicality of these approaches. Consequently, developing estimation techniques that obviate the need for payload-side sensors is crucial for enhancing the practicality and versatility of cooperative aerial systems. In response to these issues, recent research has proposed sensor-minimizing or vision-free methods that estimate the payload state using only onboard UAV measurements [12].

To address this issue, this study proposes a novel framework that estimates the payload state using only onboard UAV sensor data. This sensorless approach on the payload side reduces hardware complexity and eliminates operational constraints. A key feature of the method is that it simultaneously estimates the relative positions between UAVs and the tether attachment points on the payload, which enables the reconstruction of tether angles without requiring direct measurements. This formulation is not restricted to two UAVs and can be extended to systems involving more UAVs due to its generalized modeling of tether geometry.

The remainder of this paper is organized as follows. Section 2 describes the cooperative system and its modeling. The proposed estimation method is explained in Section 3. Then, the effectiveness of the method is validated through simulations in Section 4 and experiments in Section 5. Finally, Section 6 concludes the paper and discusses future work.

*Email address(es): suzuki-s@chiba-u.jp

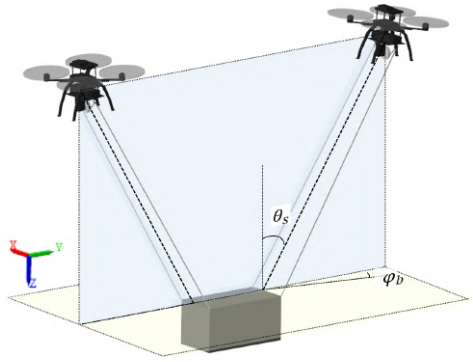


Figure 1: Cooperative manipulation system

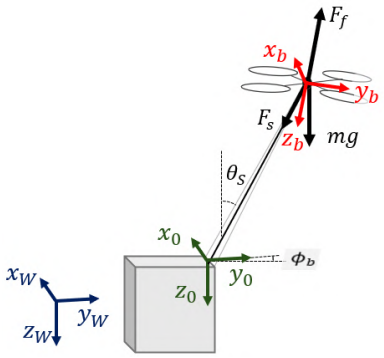


Figure 2: Definition of variables and coordinate system

2 Cooperative Manipulation System

2.1 System Configuration

This study applies the cooperative manipulation system proposed by Takemura et al. [13]. The system, shown in Figure 1, consists of two UAVs and a payload connected by a total of four tethers at its four vertices. The UAVs are quadrotors, an underactuated system in which rotation and translation are coupled. Tilting the payload in the pitch direction requires the UAV itself to tilt, causing unintended translational motion. To address this , this system is equipped with a winch mechanism on the UAV, allowing for variable tether lengths. The winch mechanism is driven by a single servo motor, where one tether extends as the other retracts. By independently adjusting tether lengths, control of the payload's pitch is achieved.

The orientation of each tether in three-dimensional space, which defines the system state, can be uniquely represented by two angles, similar to spherical coordinates: the Inclination Angle (θ_s), tilt from the vertical, and the Body-Relative Azimuth (ϕ_b), horizontal direction relative to the UAV body frame. The objective of this study is to estimate these two angles using only onboard sensors mounted on each UAV.

2.2 Modeling

The system configuration and coordinate frames are illustrated in Figure 2. The world frame ($\mathcal{F}_W$) follows the NED convention, with z_W pointing downward. The standard basis vectors of this frame are denoted as $\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z$, where $\boldsymbol{e}_z = [0, 0, 1]^T$ corresponds to the direction of gravity, and the gravitational vector is given by $\boldsymbol{g} = g\boldsymbol{e}_z$. The system also uses two additional reference frames:

- Payload frame ($\mathcal{F}_0$): located at the midpoint between the two anchor points connecting a single UAV to the payload, serving as the reference for tether orientation.
- Body-fixed frame ($\mathcal{F}_B$): attached to each UAV's center of gravity.

The azimuth is represented by the body-relative angle ϕ_b, measured from y_b toward x_b in the x_b–y_b plane. This angle is used for relative tether dynamics. For use in the world-frame dynamics, this is converted to an absolute azimuth angle ϕ_{abs}. The UAV's yaw angle ψ_{uav} and the payload's yaw angle ψ_{pay} are approximated as equal ($\psi_{pay} \approx \psi_{uav}$) due to the close coupling of the system. The relationship between these angles, illustrated in the top-down view in Figure 3, is computed as shown in Equation (1):

$$\phi_{abs} = (\psi_{uav} + \phi_b) + \frac{\pi}{2} \tag{1}$$

where the $\pi/2$ term adjusts the North-referenced yaw angle ψ_{uav} to align with the East-based azimuth convention. The dynamics follow a spherical pendulum with an accelerating pivot. Depending on whether relative or global angular rates are required, either ϕ_b or ϕ_{abs} is used in the system dynamics.

The tether motion is described using a spherical coordinate system. The elevation angle θ_s, is measured from the local vertical. The azimuth angle ϕ_{abs}, is measured from the North-East direction. The system is characterized by the orthogonal unit vectors $\boldsymbol{u}_r, \boldsymbol{u}_\theta, \boldsymbol{u}_\phi\}$, defined in Equations (2) - (4):

$$\boldsymbol{u}_r = \begin{bmatrix} \sin\theta_s \cos\phi_{abs} \\ \sin\theta_s \sin\phi_{abs} \\ -\cos\theta_s \end{bmatrix} \tag{2}$$

$$\boldsymbol{u}_\theta = \begin{bmatrix} \cos\theta_s \cos\phi_{abs} \\ \cos\theta_s \sin\phi_{abs} \\ -\sin\theta_s \end{bmatrix} \tag{3}$$

$$\boldsymbol{u}_\phi = \begin{bmatrix} -\sin\phi_{abs} \\ \cos\phi_{abs} \\ 0 \end{bmatrix} \tag{4}$$

The UAV's translational dynamics are described in the world frame $\mathcal{F}_W$ by Equation (5) :

$$\ddot{\boldsymbol{p}} = \frac{1}{m}\boldsymbol{R}_f \boldsymbol{F}_f - \frac{1}{m}\boldsymbol{F}_s + \boldsymbol{g} \tag{5}$$

Here, $\ddot{\boldsymbol{p}}$ is the acceleration of the UAV in the world frame $\mathcal{F}_W$. The right-hand side accounts for the total force acting on the

http://www.imavs.org/

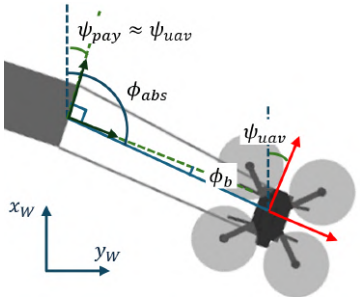


Figure 3: Top-down view of the azimuth angle definitions

UAV, expressed in the world frame. This includes the body-frame thrust $\boldsymbol{F}_f$ (which is transformed into the world frame by the rotation matrix $\boldsymbol{R}_f$), the world-frame tether force $\boldsymbol{F}_s$, and the gravity vector $\boldsymbol{g}$.

The equations of motion for the tether angles are derived from Newton's second law applied to the payload. The model accounts for gravity, tether tension, and the inertial effects of the UAV's acceleration, which is approximated by its thrust. The resulting vector equation is projected onto the spherical coordinate basis vectors ($\boldsymbol{u}_\theta$, $\boldsymbol{u}_\phi$) to yield the component-wise dynamics shown in Equations (6) and (7):

$$\ddot{\theta}_s = -\frac{g}{l}\sin\theta_s + \frac{\boldsymbol{R}_f\boldsymbol{F}_f \cdot \boldsymbol{u}_\theta}{ml} + \dot{\phi}_{abs}^2 \sin\theta_s \cos\theta_s \quad (6)$$

$$\ddot{\phi}_{abs} = \frac{\boldsymbol{R}_f\boldsymbol{F}_f \cdot \boldsymbol{u}_\phi}{ml\sin\theta_s} - 2\dot{\theta}_s\dot{\phi}_{abs}\cot\theta_s \quad (7)$$

To estimate the tether angles for the UKF's observation step, we use the UAV's IMU measurements. An on-board accelerometer measures the specific force, which is the total non-gravitational acceleration. From Equation (5), the specific force expressed in the world frame is $(\ddot{\boldsymbol{p}} - \boldsymbol{g})$. Since the IMU provides this measurement in the UAV's body frame ($\mathcal{F}_B$), we must transform this vector from the world frame to the body frame by multiplying it by the rotation matrix $\boldsymbol{R}_f^T$. Thus, the accelerometer output $\boldsymbol{a}_b$ is modeled as Equation (8):

$$\boldsymbol{a}_b = \frac{1}{m}\left(\boldsymbol{F}_f - \boldsymbol{R}_f^T\,\boldsymbol{F}_s\right) \quad (8)$$

To find the tether orientation, we first estimate the tension force vector $\boldsymbol{F}_s$. Rearranging Equation (8) isolates the tether force $\boldsymbol{F}_s$, which in the world frame can be expressed using the known thrust command $\boldsymbol{F}_f$ and measured acceleration $\boldsymbol{a}_b$, as shown in Equation (9):

$$\boldsymbol{F}_s = \boldsymbol{R}_f(\boldsymbol{F}_f - m\boldsymbol{a}_b) \quad (9)$$

The inclination angle θ_s' is then calculated as the angle between the estimated tension vector $\boldsymbol{F}_s$ and the vertical axis $\boldsymbol{e}_z$. According to the geometric definition of the dot product, the cosine of this angle is the dot product of the two vectors divided by the product of their magnitudes. Since $\boldsymbol{e}_z$ is a unit vector ($\|\boldsymbol{e}_z\| = 1$), we can solve for the angle θ_s' by taking the arccosine:

$$\theta_s' = \arccos\left(\frac{\boldsymbol{F}_s \cdot \boldsymbol{e}_z}{\|\boldsymbol{F}_s\|}\right) \quad (10)$$

Then, the body-relative azimuth angle ϕ_b' is computed as detailed in Equation (11). This is achieved by first calculating the world-frame azimuth from the components of the tension vector $\boldsymbol{F}_s$, and then converting it to the body-relative frame by subtracting the UAV's yaw angle ψ_{uav} and compensating for the coordinate system offset of $\frac{\pi}{2}$, consistent with the convention established in Equation (1):

$$\phi_b' = \operatorname{atan2}\left((\boldsymbol{F}_s)_y, (\boldsymbol{F}_s)_x\right) - \psi_{uav} - \frac{\pi}{2} \quad (11)$$

These estimated angles θ_s' and ϕ_b' are used as observation values in the UKF.

3 State Estimation Method

3.1 Tether Angle Estimation

The following subsections describe in detail the system models used for the UKF, which, unlike the Extended Kalman Filter (EKF), does not require linearization and is therefore well suited for highly nonlinear systems such as the tethered UAV dynamics.

The estimated state of the system, $\hat{\boldsymbol{x}}$, at a discrete time k is defined as follows in Equation (12):

$$\hat{\boldsymbol{x}}^k = \begin{bmatrix}\hat{\theta}_s^k & \hat{\dot{\theta}}_s^k & \hat{\phi}_b^k & \hat{\dot{\phi}}_b^k\end{bmatrix}^T \quad (12)$$

The state transition is modeled by the discrete Wiener process acceleration model shown in Equation (13). This model integrates the state derivatives using the fourth-order Runge-Kutta method (RK_4) [14, 15].

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \mathrm{RK}_4(\boldsymbol{f}(\boldsymbol{x}^k)) + \boldsymbol{\Gamma}\boldsymbol{v}^k \quad (13)$$

where $\boldsymbol{f}(\boldsymbol{x}^k)$ is the state derivative vector, and $\boldsymbol{\Gamma}\boldsymbol{v}^k$ is the process noise term.

The vector $\boldsymbol{f}(\boldsymbol{x}^k)$ is derived by adapting the full physical dynamics from Equations (6) and (7) for implementation in the UKF. To improve computational efficiency and because the effects of these rotational forces are considered negligible for this application, the model is first simplified by neglecting the centrifugal and Coriolis force terms.

The model then incorporates the inertial force $(-m\boldsymbol{a}_0^k)$ resulting from the payload's acceleration, $\boldsymbol{a}_0$. This acceleration is not measured directly and is instead estimated by taking the UAV's absolute acceleration, $\boldsymbol{a}_d$, and subtracting the tether's relative acceleration from the previous step, $\boldsymbol{a}_{rel}(k-1)$, as shown in Equation (14):

$$\boldsymbol{a}_0(k) \approx \boldsymbol{a}_d(k) - \boldsymbol{a}_{rel}(k-1) \quad (14)$$

http://www.imavs.org/

Here, the UAV's absolute acceleration is $\boldsymbol{a}_d = \boldsymbol{R}_f \boldsymbol{a}_b + \boldsymbol{g}$. The relative acceleration, $\boldsymbol{a}_{rel}$, is reconstructed from the estimated state using its spherical coordinate components, as defined in Equations (15) and (16):

$$\boldsymbol{a}_{rel} = a_r \boldsymbol{u}_r + a_\theta \boldsymbol{u}_\theta + a_\phi \boldsymbol{u}_\phi \tag{15}$$

$$a_r = -l\dot{\theta}_s^2, \quad a_\theta = l\ddot{\theta}_s, \quad a_\phi = l\ddot{\phi}_{abs} \sin\theta_s \tag{16}$$

Finally, the azimuth dynamics are converted to the body-relative frame used in the state vector. The state vector uses the body-relative azimuth ϕ_b instead of the absolute azimuth ϕ_{abs} because the UKF is formulated in the UAV body frame, and the relative angle directly relates to the tether's motion with respect to the UAV. Therefore, the body-relative angular acceleration is computed by subtracting the UAV's yaw acceleration from the absolute angular acceleration ($\ddot{\phi}_b = \ddot{\phi}_{abs} - \ddot{\psi}_{uav}$).

By incorporating these three modifications, the final state derivative vector $\boldsymbol{f}(\boldsymbol{x}^k)$ is constructed as shown in Equation (17):

$$\boldsymbol{f}(\boldsymbol{x}^k) = \begin{bmatrix} \dot{\hat{\theta}}_s^k \\ -\frac{g}{l}\sin\hat{\theta}_s^k + \frac{(\boldsymbol{R}_f \boldsymbol{F}_f - m\boldsymbol{a}_0^k)\cdot \boldsymbol{u}_\theta^k}{ml} \\ \dot{\hat{\phi}}_b^k \\ \frac{(\boldsymbol{R}_f \boldsymbol{F}_f - m\boldsymbol{a}_0^k)\cdot \boldsymbol{u}_\phi^k}{ml\sin\hat{\theta}_s^k} - \ddot{\psi}_{uav}^k \end{bmatrix} \tag{17}$$

Finally, the process noise model is defined. We assume two independent noise sources affecting the inclination and azimuth accelerations, respectively. These are represented by a zero-mean Gaussian noise vector $\boldsymbol{v}^k$, defined in Equation (18):

$$\boldsymbol{v}^k = \begin{bmatrix} v_\theta^k \\ v_\phi^k \end{bmatrix} \tag{18}$$

The covariance of this input noise is given by the diagonal matrix $\boldsymbol{\Sigma}$ in Equation (19). Its diagonal elements, σ_θ^2 and σ_ϕ^2, are the variances of the respective noise components v_θ^k and v_ϕ^k.

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_\theta^2 & 0 \\ 0 & \sigma_\phi^2 \end{bmatrix} \tag{19}$$

The noise distribution matrix $\boldsymbol{\Gamma}$, which maps the input noise vector $\boldsymbol{v}^k$ to the state space. Its terms, T_s and $T_s^2/2$, result from integrating acceleration noise into velocity and angle. It is defined in Equation (20):

$$\boldsymbol{\Gamma} = \begin{bmatrix} T_s^2/2 & 0 \\ T_s & 0 \\ 0 & T_s^2/2 \\ 0 & T_s \end{bmatrix} \tag{20}$$

The final process noise covariance matrix $\boldsymbol{Q}$ is then constructed by combining the distribution matrix $\boldsymbol{\Gamma}$ from Equation (20) and the input noise covariance $\boldsymbol{\Sigma}$ from Equation (19) using the sandwich product, as shown in Equation (21):

$$\boldsymbol{Q} = \boldsymbol{\Gamma}\boldsymbol{\Sigma}\boldsymbol{\Gamma}^T \tag{21}$$

In the UKF, the process noise covariance $\boldsymbol{Q}$ is added during prediction. The Kalman gain is then computed as $\boldsymbol{K} = \boldsymbol{P}xy\boldsymbol{P}yy^{-1}$, where $\boldsymbol{P}xy$ denotes the cross-covariance between state and measurement, and $\boldsymbol{P}yy$ is the predicted measurement covariance. This gain blends the predicted state with measurements, and the updated covariance $\boldsymbol{P}$ reflects the reduced estimation uncertainty.

3.2 Geometric Payload Pose Estimation

This section describes the method for estimating the 6-DOF (degrees of freedom) state of the payload using geometric relationships. The estimation relies primarily on the tether angle states for each UAV, as obtained from the UKF.

Payload Position ($\boldsymbol{p}_p$) First, the 3D position of each payload anchor point, $\boldsymbol{p}_{pa,i}$, is calculated. The tether's unit direction vector, $\hat{\boldsymbol{r}}_i$, is reconstructed from the estimated angles ($\hat{\theta}_{s,i}$, $\hat{\phi}_{b,i}$) and the UAV's yaw ($\psi_{uav,i}$), consistent with the definitions in Equations (1) and (2). The anchor point position is then found via forward kinematics as shown in Equation (22):

$$\boldsymbol{p}_{pa,i} = \boldsymbol{p}_{ua,i} + l_i \boldsymbol{u}_{r,i} \tag{22}$$

where $\boldsymbol{p}_{ua,i}$ is the UAV's anchor position. The payload's center, $\boldsymbol{p}_p$, is then calculated as the midpoint of these two points in Equation (23):

$$\boldsymbol{p}_p = \frac{\boldsymbol{p}_{pa,1} + \boldsymbol{p}_{pa,2}}{2} \tag{23}$$

Roll Angle (φ_p) The payload's roll angle, φ_p, is determined by the angle of the vector connecting the two anchor points ($\boldsymbol{p}_{pa,2} - \boldsymbol{p}_{pa,1}$) when projected onto the world frame's y_W–z_W plane, as calculated in Equation (24):

$$\varphi_p = \text{atan2}\left((\boldsymbol{p}_{pa,2} - \boldsymbol{p}_{pa,1})_z, (\boldsymbol{p}_{pa,2} - \boldsymbol{p}_{pa,1})_y\right) \tag{24}$$

Pitch Angle (θ_p) The pitch angle, θ_p, is controlled by the winch mechanism (Section 2). It is therefore estimated directly from a pre-calibrated function of the winch encoder measurements, as shown conceptually in Equation (25):

$$\theta_p = f_{\text{winch}}(l_{\text{winch},1}, l_{\text{winch},2}) \tag{25}$$

Yaw Angle (ψ_p) The payload's yaw, ψ_p, is assumed to align with the average horizontal direction of the tethers. First, the absolute compass heading of each tether, $\phi_{r,i}$, is calculated in Equation (26):

$$\phi_{r,i} = \psi_{uav,i} + \phi_{b,i} \tag{26}$$

The payload's yaw is then estimated as the circular average of these two headings to correctly handle angle wrapping, as shown in Equation (27):

$$\psi_p = \text{atan2}\left(\sin\phi_{r,1} + \sin\phi_{r,2}, \cos\phi_{r,1} + \cos\phi_{r,2}\right) \tag{27}$$

This completes the geometric estimation of the payload's 6-DOF pose.

http://www.imavs.org/

http://www.imavs.org/

4 Verification with Simulation

To validate the proposed state estimator, a simulation environment was constructed using MATLAB/Simulink and Simscape Multibody. The physical and filter design parameters used in the simulation are summarized in Table 1.

Reference trajectories for the payload were predefined, and UAV trajectories were computed by inverse kinematics and tracked using local PID control, making the payload motion purely feedforward-driven. The UKF was run offline using logged simulation data. Sensor noise was excluded; robustness will be evaluated in real experiments.

Attitude and position maneuvers were performed, and the logged data provided inputs for the UKF and ground-truth tether angles for evaluation.

4.1 Estimation During Attitude Maneuvers

To evaluate the estimation performance, the payload was commanded to perform a series of attitude maneuvers. Based on this simulation data, the proposed tether angle estimation was conducted offline. The results are presented in Figure 4, which shows the estimated inclination θ_s and azimuth ϕ_b angles for the tethers of each UAV.

The inclination angle θ_s was estimated with high accuracy throughout the maneuver. For the azimuth angle ϕ_b, although the overall estimation is acceptable, transient errors are observed at the beginning of attitude changes. These discrepancies are attributed to the inertial effects neglected in the simplified dynamics model. However, since the estimation converges shortly afterward, the estimator is considered effective during attitude maneuvers. The payload's attitude, calculated from these tether angle estimates, was compared to the ground truth (Figure 5). The high-precision tracking confirms the effectiveness of the proposed method and validates the underlying tether angle estimator.

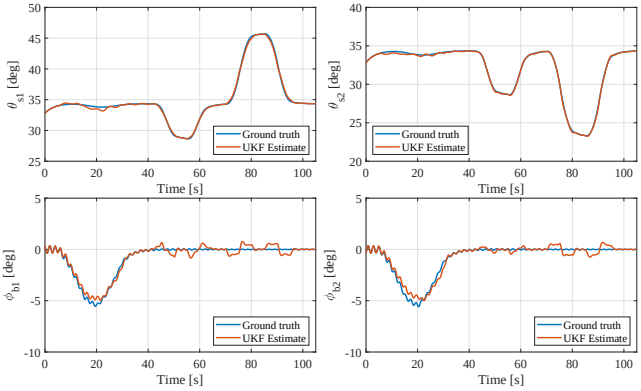
4.2 Estimation During Position Maneuvers

The performance during payload position maneuvers is shown in Figure 6. The estimated azimuth angle ϕ_b captured the payload's pendulum-like swing, seen as x-axis oscillations in Figure 7. The inclination angle θ_s is mostly accurate, but errors appear during vertical (z-axis) motion. This discrepancy is likely due to unmodeled dynamics at the payload-side tether attachment points. Furthermore, the payload's position was calculated from these tether estimates, and the result is compared to the ground truth in Figure 7. Since the estimation errors in the tether angles were minor, the resulting payload position also tracks the true trajectory with high accuracy. This confirms that the proposed estimation method is sufficiently precise for payload state tracking during position maneuvers.

Notably, the estimation of ϕ_b is more accurate here than during attitude maneuvers. This is because the primarily translational motion during position maneuvers aligns better with the model's assumptions than the complex rotational motion seen during attitude maneuvers.

Physical Parameters	
Payload size	$0.2\text{m} \times 0.2\text{m} \times 0.4\text{m}$
Payload mass	0.5kg
UAV mass	1.64kg
Tether length	1.2m
UKF Design Parameters	
Proc. noise std. (σ_θ)	1.0×10^{-3}
Proc. noise std. (σ_ϕ)	5.0×10^{-4}
Obs. noise cov. ($\boldsymbol{R}$)	$\text{diag}\left(1.0 \times 10^{-3}, 1.0 \times 10^{-3}\right)$

Table 1: Parameters used in the simulation



(a) Tether angles: UAV1 (b) Tether angles: UAV2

Figure 4: Tether angle estimation during attitude maneuvers

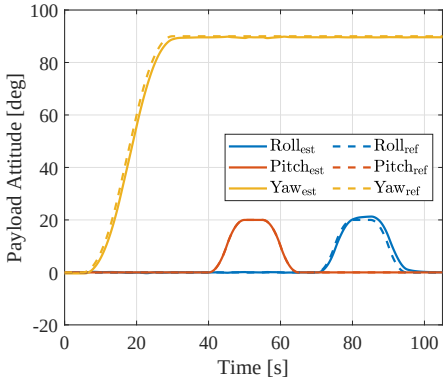


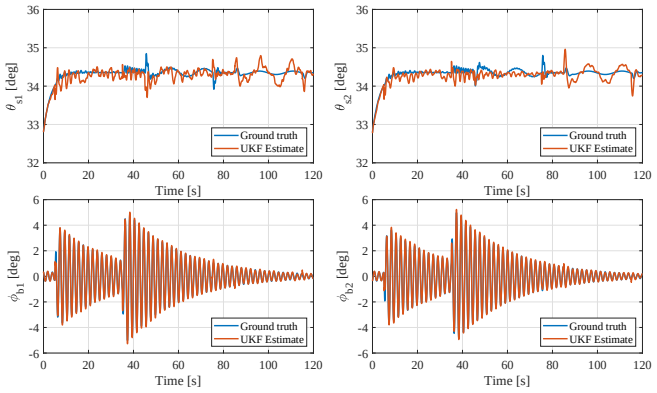
Figure 5: Estimation result of payload attitude

5 Verification with Experiment

5.1 Experimental Setup

Flight experiments were conducted to verify the proposed estimation method in a real-world environment. The experiments aimed to evaluate its accuracy under sensor noise and environmental disturbances.

The experiments were conducted in a Motion Capture (MoCap) environment with two quadrotors carrying a pay-



(a) Tether angles: UAV1 (b) Tether angles: UAV2

Figure 6: Tether angle estimation during position maneuvers

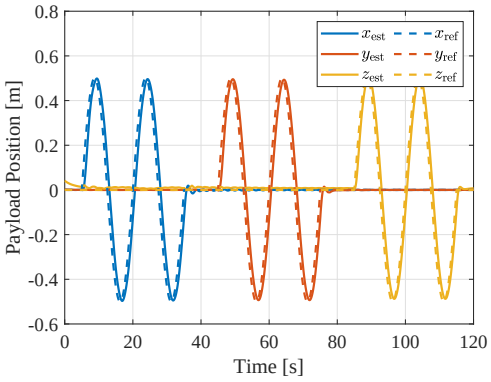


Figure 7: Estimation result of payload position

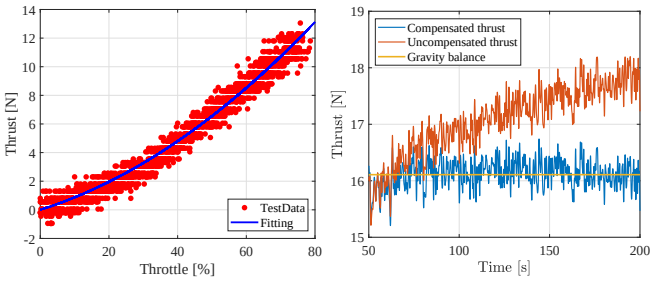
load. The MoCap system provided ground-truth data for evaluation and position feedback for the UAVs, which followed predefined trajectories. UAV synchronization was achieved through a centralized control system using ROS(Robot Operating System) over Wi-Fi. While the physical parameters were identical to the simulation (Table 1), the UKF design parameters were empirically tuned to account for the differing sensor noise and unmodeled disturbances present in the real experiment. The final values are summarized in Table 2.

5.2 *Thrust Estimation Model and Voltage Compensation*

The accuracy of the proposed estimation method depends on an accurate thrust value $\boldsymbol{F}_f$ as an input to the prediction model (Equations (6) and (7)) and the observation model (Equation (9)). However, direct in-flight measurement of

UKF Design Parameters	
Proc. noise std. (σ_θ)	2.5×10^{-2}
Proc. noise std. (σ_ϕ)	1.0×10^{-2}
Obs. noise cov. ($\boldsymbol{R}$)	diag $(0.25, 0.20)$

Table 2: UKF parameters tuned for the experiment



(a) Static thrust test result (b) Voltage-compensated thrust

Figure 8: Experimental analysis of thrust estimation

thrust is infeasible. While prior work has treated thrust as a constant value [14], this assumption is not valid for our dynamic maneuvers and would degrade estimation accuracy. Therefore, we adopted a model that compensates for the battery voltage drop in-flight. This involved a two-step process.

First, a static thrust test was conducted to establish a baseline thrust-throttle relationship, shown in Figure 8a. The result was approximated by Equation (28):

$$\|\boldsymbol{F}_f\| = 0.0011x^2 + 0.76x \tag{28}$$

where $\|\boldsymbol{F}_f\|$ is the generated thrust [N], and x is the throttle percentage [%].

Second, to compensate for battery voltage drop, this baseline model was adjusted in real-time using the measured voltage V, as shown in Equation (29):

$$\|\boldsymbol{F}_f\| = 0.8\left(\frac{V}{16.0}\right)^2 (0.0011x^2 + 0.76x) \tag{29}$$

where V is the measured battery voltage, $V_0 = 16.0$ V is the nominal voltage used in the calibration, and the factor 0.8 reflects the saturation at 80% throttle, introduced as a safety margin to prevent the propulsion system from exceeding its current rating.

The validation was performed in hover, where the thrust can be directly inferred from the equilibrium of forces, providing a reliable baseline for evaluating the model. The estimated thrust from both the original and compensated equations was compared with the gravitational force. The comparison in Figure 8b shows the compensated model more accurately captures the true thrust required for hovering, validating the voltage compensation approach.

5.3 *Experimental Results and Discussion*

An attitude maneuver experiment was conducted, as shown in Figure 9a. The resulting payload attitude is presented in Figure 9b. Figures 9c and 9d show the accelerations of the UAVs, while Figure 10 shows the thrust from the voltage-compensated model (Equation (29)).

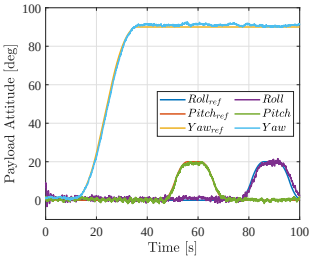
The tether angles and 6-DOF payload state were estimated from the flight data, as shown in Figure 11 and Figure 12. A quantitative evaluation is summarized in Table 3.

http://www.imavs.org/

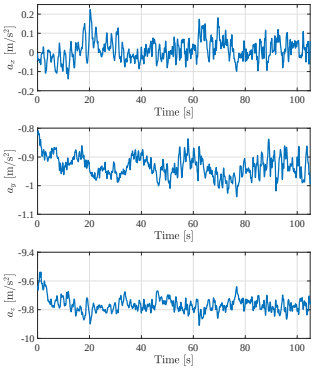
http://www.imavs.org/



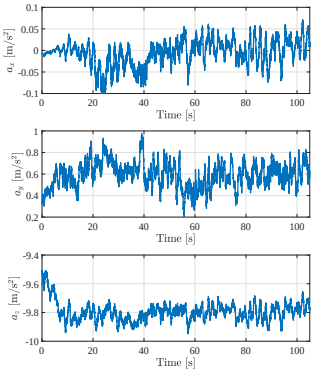
(a) Overview of the experiment



(b) Result of payload attitude



(c) UAV1 acceleration



(d) UAV2 acceleration

Figure 9: Flight test for payload attitude maneuver

The results confirm the proposed framework is effective, achieving attitude RMSE within a few degrees and position RMSE within several centimeters.

The primary performance limitation observed was high-frequency oscillations in the estimates, a phenomenon not present in the simulation. These oscillations can be traced directly to high-frequency noise in the input signals, such as the measured UAV accelerations (Figures 9c and 9d) and the thrust estimate (Figure 10). This noise propagates through the model, causing oscillations in the final estimates. This effect was most pronounced in the lateral and rotational axes, where the Y-axis exhibited the largest position RMSE (4.30 cm), alongside significant errors in Roll and Yaw. In contrast, the Pitch estimate remained highly accurate with an RMSE of only 0.76 deg. This accuracy stems from its winch-based model, independent of the noisy IMU and thrust inputs, reinforcing the conclusion that input signal quality is the dominant error source for the other states. However, this simplified model has its own limitations, as it did not fully capture the minor pitching oscillations caused by the payload's natural swing from aerodynamic effects. Additionally, the omission of centrifugal and Coriolis terms, though less significant than sensor noise, remains a modeling limitation for more dynamic maneuvers.

State	RMSE	Max Error
Attitude Errors [deg]		
Roll	2.392	8.363
Pitch	0.760	2.108
Yaw	2.034	6.811
Position Errors [m]		
X	0.0271	0.1230
Y	0.0430	0.1328
Z	0.0258	0.0717

Table 3: RMSE and maximum error of payload estimation

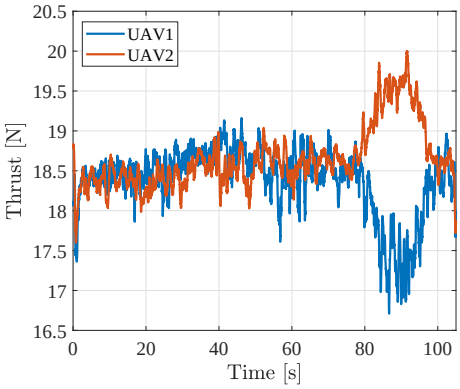
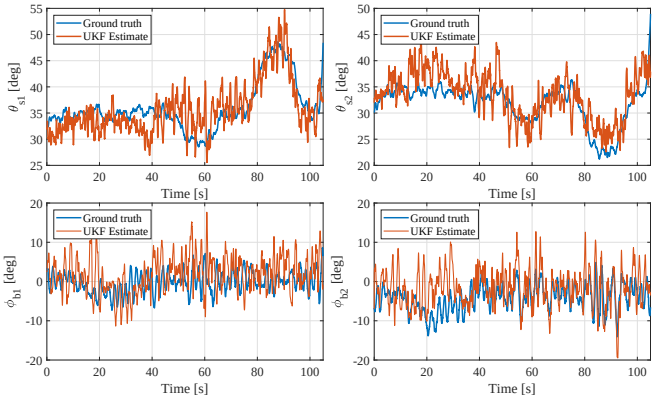


Figure 10: Thrust calculations during experiment



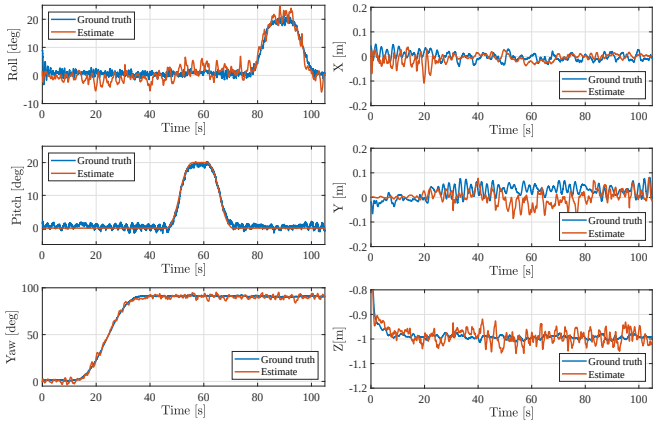
(a) Tether angles: UAV1 (b) Tether angles: UAV2

Figure 11: Tether angle estimation in experiment

In summary, this experiment validates the proposed strategy under real-world conditions and identifies input signal quality as the main performance limitation. Future work should therefore focus on improved input filtering and more robust dynamic modeling for practical deployment.

6 Conclusion

This study developed and validated a UKF-based estimation framework for a cooperative manipulation system. We successfully demonstrated that the payload's 6-DOF state can be determined with practical accuracy using only each UAV's onboard IMU and position data. The method's effectiveness



(a) Estimated payload attitude (b) Estimated payload position

Figure 12: Payload state estimation in experiment

was confirmed through both dynamic simulations and real-world flight experiments, establishing a baseline for sensor-based payload state awareness in such systems.

The experimental validation highlighted the primary challenges for real-world deployment: performance degradation due to unmodeled dynamics, such as aerodynamic effects, and the propagation of sensor noise. A key challenge for real-time implementation is filter divergence, which requires robust modeling and recovery strategies. Future work will refine the dynamic model and implement the estimator in a real-time control loop. Successfully tackling these challenges is a critical step toward the practical deployment of autonomous and dynamic cooperative manipulation systems.

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] Miguel Ángel Trujillo, José Ramiro Martínez-de Dios, Carlos Martín, Antidio Viguria, and Aníbal Ollero. Novel aerial manipulator for accurate and robust industrial ndt contact inspection: A new tool for the oil and gas inspection industry. *Sensors*, 19(6), 2019.

[3] Ivana Palunko, Rafael Fierro, and Patricio Cruz. Trajectory generation for swing-free maneuvers of a quadrotor with suspended payload: A dynamic programming approach. In *2012 IEEE International Conference on Robotics and Automation*, pages 2691–2697, 2012.

[4] Stamatina C. Barakou, Costas S. Tzafestas, and Kimon P. Valavanis. A survey of modeling and control approaches for cooperative aerial manipulation. In *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 610–617, 2024.

[5] Koushil Sreenath and Vijay Kumar. Dynamics, control and planning for cooperative manipulation of payloads suspended by cables from multiple quadrotor robots. In *Robotics: Science and Systems*, 2013.

[6] Kristian Klausen, Chris Meissen, Thor I. Fossen, Murat Arcak, and Tor Arne Johansen. Cooperative control for multirotors transporting an unknown suspended load under environmental disturbances. *IEEE Transactions on Control Systems Technology*, 28(2):653–660, 2020.

[7] Dario Sanalitro, Heitor J. Savino, Marco Tognon, Juan Cortés, and Antonio Franchi. Full-pose manipulation control of a cable-suspended load with multiple uavs under uncertainties. *IEEE Robotics and Automation Letters*, 5(2):2185–2191, 2020.

[8] Junyi Geng and Jack W. Langelaan. Cooperative transport of a slung load using load-leading control. *Journal of Guidance, Control, and Dynamics*, 43(7), 2020.

[9] Shubham Barawkar and K. Kumar. Force-torque (f-t) based multi-drone cooperative transport using fuzzy logic and low-cost and imprecise f-t sensor. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 237(11):2517–2544, 2023.

[10] Yasheng Li, Sifan Ge, and Giuseppe Loianno. Cooperative transportation of cable suspended payloads with mavs using monocular vision and inertial sensing. *IEEE Robotics and Automation Letters*, 6(3):516–523, 2021.

[11] Sara Tang, Thomas Wüest, and Vijay Kumar. Aggressive flight with suspended payloads using vision-based control. *IEEE Robotics and Automation Letters*, 3(4):3410–3417, 2018.

[12] Heng Xie, Kaixu Dong, and Pakpong Chirarattananon. Cooperative transport of a suspended payload via two aerial robots with inertial sensing. *IEEE Access*, 10:81764–81776, 2022.

[13] Yuichi Takemura, Yuki Ohira, and Satoshi Suzuki. Reserch on cooperative manipulation by multiple drones with winch mechanisms. In *Proceedings of the 42nd Annual Conference of the Robotics Society of Japan (RSJ2024)*, pages 3J4–01, 2024. (In Japanese).

[14] Sergei Lupashin and Raffaello D'Andrea. Stabilization of a flying vehicle on a taut tether using inertial sensing. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2432–2438, 2013.

[15] Yaakov Bar-Shalom, X. Rong Li, and Thia Kirubarajan. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., New York, NY, 2002.

http://www.imavs.org/

Spiking Neural Networks for High-Speed Continuous Quadcopter Control Using Proximal Policy Optimization

M.F. van Breukelen Castillo[*], R. Ferede, R.W. Vos, C. De Wagter G.C.H.E. de Croon [†]
Delft University of Technology, Kluyverweg 1, 2629 HS Delft, The Netherlands

http://www.imavs.org/

ABSTRACT

We present the first demonstration of a fully spiking actor-critic neural network policy, trained via Proximal Policy Optimization (PPO), for continuous control of an agile high-speed quadcopter in a gate-based navigation task. The spiking neural network (SNN) controller employs Leaky Integrate-and-Fire neurons with surrogate gradient training and spike-rate decoding over multiple integration cycles, and it is benchmarked against a comparable artificial neural network (ANN) controller in both simulation and real-world flight tests. Results show that despite being trained to the same reward level, the SNN achieves superior performance in simulation, achieving higher episode rewards, greater robustness and reduced crash rate. Additionally, in 12-second real-world trials, the SNN outperforms the ANN, attaining a higher average reward (70.63 vs 59.77), greater mean velocity (7.94 vs 6.99 m/s), and more gates cleared (46.33 vs 40.67). An analysis of the spike integration cycle count reveals a clear trade-off: lower cycle counts (fewer integration steps per control update) reduce control output resolution and hinder learning, whereas higher cycle counts improve smoothness but increase inference latency. Moderate cycle counts (5 or 8) provide the best balance, yielding high rewards, smoother outputs, and low execution time overhead. These findings represent a key step forward for neuromorphic control in embedded autonomous systems, demonstrating that SNN-based policies can outperform conventional ANN controllers in high-speed, agile robotic tasks.

1 INTRODUCTION

Over the past two decades, machine learning has greatly advanced autonomous systems, from smartphones and self-driving vehicles to large language models. This progress comes at a cost: deep networks are energy-intensive, demanding substantial resources for training and inference. As their use grows, there is a pressing need for energy-efficient models that can deliver real-time performance on embedded systems. This challenge is acute in micro aerial vehicles (MAVs), which require fast, low-latency control under strict power and hardware limits. Tasks such as drone racing or agile navigation push the limits of onboard computation, as controllers must run at high frequencies on lightweight, battery-powered platforms. Although artificial neural networks (ANNs) have shown strong control capabilities [1, 2], their poor energy efficiency remains a critical limitation.

Spiking Neural Networks (SNNs) have emerged as a promising alternative. Inspired by the sparse, event-driven signalling of biological neurons, SNNs operate asynchronously through binary spikes generated only when a neuron's membrane potential crosses a threshold. As a result, large parts of the network remain inactive, and computation scales with the number of spikes rather than the number of neurons. This enables low-power computation with minimal latency and can yield orders-of-magnitude energy savings compared to ANNs that update all neurons continuously [3]. These properties make SNNs attractive for real-time control on embedded platforms when deployed on neuromorphic hardware such as Intel's Loihi.

Despite this promise, SNNs remain underexplored in closed-loop robotic control. Prior work has examined their use in vision [4], neuromorphic sensing [5], and motor tasks [6], but not in high-speed quadcopter control. Reinforcement learning (RL) provides a natural framework, as agents learn control policies through trial-and-error interaction with an environment to maximise long-term reward [7]. Recent methods have enabled SNNs to be trained with surrogate gradients [8, 9], yet applications have mostly been confined to low-dimensional benchmarks, and robust real-time deployment is still lacking. Initial demonstrations show progress: Paredes-Vallés et al. implemented a spiking vision-to-control pipeline on Loihi, enabling autonomous drone flight at 200 Hz with microjoule-level inference cost [10], and Stroobants et al. achieved spiking attitude control on a Crazyflie at 500 Hz [11]. These works highlight the potential of neuromorphic approaches, but none address high-speed, continuous quadcopter control directly with fully spiking networks.

In this work we present the first implementation of a fully spiking actor-critic architecture for continuous quadcopter control in an agile gate navigation task. Learning is

[*]ORCID: (0009-0001-9337-7960)
[†]Email address(es): michaelvanbreukelen509@gmail.com , {R.Ferede; R.W.Vos, C.deWagter, G.C.H.E.deCroon}@tudelft.nl
Code=https://github.com/michael2992/msc_spiking_quadcopter_control

http://www.imavs.org/

carried out with Proximal Policy Optimization (PPO), a reinforcement learning algorithm that updates the policy by sampling trajectories, evaluating performance, and making incremental changes while constraining updates to remain stable [12]. The network is built from Leaky Integrate-and-Fire (LIF) neurons, a standard spiking model in which inputs accumulate in a membrane potential that decays over time and trigger a spike when the threshold is crossed. To obtain continuous commands from discrete spikes, we apply spike-rate decoding, where spike rates are calculated over a short window and mapped to motor actions. The SNN architecture is matched in structure and parameter count to an ANN baseline [13], ensuring fair comparison.

Our results show that the SNN not only achieves strong performance in simulation but also outperforms the ANN in real-world flight tests, demonstrating higher robustness, greater average velocity, and more gates traversed within a 12-second interval. We also analyse how the number of spike-integration cycles affects control performance and latency. This work bridges the gap between neuromorphic computing and reinforcement learning, providing a foundation for SNN-based control policies on energy-constrained aerial robots. The code for the training, simulation, and flight data analysis for the project is available at: `https://github.com/michael2992/msc_spiking_quadcopter_control`, and the videos of the real flight tests are available at: `https://drive.google.com/drive/folders/1uGINGe71wu0Hrh0_ZBDAHMIO-s6B5qd3?usp=drive_link`.

2 METHODOLOGY

This section outlines the quadcopter model, simulation environment, reinforcement learning framework and spiking network adaptations used for developing and evaluating the SNN controller. Our approach builds directly on the environment and task setup from [14] and [13], but is distinct in that it focuses exclusively on the 5-inch quadcopter platform with a focus on training the spiking actor-critic policy.

2.1 Quadcopter Model

We adopt the parametric quadcopter dynamics model from [13] defined in continuous time. The state vector $\mathbf{x}$ and input vector $\mathbf{u}$ are defined as:

$$\mathbf{x} = [\mathbf{p}, \mathbf{v}, \boldsymbol{\lambda}, \boldsymbol{\Omega}, \boldsymbol{\omega}]^T, \quad \mathbf{u} = [u_1, u_2, u_3, u_4]^T \in [0,1]^4 \quad (1)$$

Here, $p \in \mathbb{R}^3$ is position, $v \in \mathbb{R}^3$ is velocity, $\lambda \in \mathbb{R}^3$ represents Euler angles, $\Omega \in \mathbb{R}^3$ the body rates, and ω_i the propeller speeds in rad/s. The control input $\mathbf{u} \in [0,1]^4$ represents the normalized motor commands. The equations of motion are given by:

$$\dot{\mathbf{p}} = \mathbf{v}, \qquad \dot{\mathbf{v}} = g\mathbf{e}_3 + R(\lambda)\mathbf{F} \quad (2)$$

$$\dot{\lambda} = Q(\lambda)\boldsymbol{\Omega}, \qquad \dot{\boldsymbol{\Omega}} = \mathbf{M} \quad (3)$$

$$\dot{\omega}_i = \frac{\omega_{ci} - \omega_i}{\tau} \quad (4)$$

Where $R(\lambda)$ is the rotation matrix and $Q(\lambda)$ the transformation matrices from body rates to Euler angle derivatives. The motor steady-state response is modelled as:

$$\omega_{ci} = (\omega_{\max} - \omega_{\min})\sqrt{k_l u_i^2 + (1-k_l)u_i} + \omega_{\min} \quad (5)$$

The specific thrust force F and moment M acting on the body are given by:

$$\mathbf{F} = \left[-\sum_{i=1}^{4} k_x v_x^B \omega_i, \; -\sum_{i=1}^{4} k_y v_y^B \omega_i, \; -\sum_{i=1}^{4} k_\omega \omega_i^2 \right]^T \quad (6)$$

$$\mathbf{M} = \begin{bmatrix} -k_{p1}\omega_1^2 - k_{p2}\omega_2^2 + k_{p3}\omega_3^2 + k_{p4}\omega_4^2 \\ -k_{q1}\omega_1^2 + k_{q2}\omega_2^2 - k_{q3}\omega_3^2 + k_{q4}\omega_4^2 \\ -k_{r1}\omega_1 + k_{r2}\omega_2 + k_{r3}\omega_3 - k_{r4}\omega_4 + \\ \ldots - k_{r5}\omega_1 + k_{r6}\omega_2 + k_{r7}\omega_3 - k_{r8}\omega_4 \end{bmatrix} \quad (7)$$

All parameters k_*, $\omega_{\min}$, and $\omega_{\max}$ are identified for the 5-inch drone following [13].

2.2 Reinforcement Learning Task

The control objective is to autonomously navigate a 5-inch racing quadcopter through a sequence of 7 square gates arranged in a figure-eight pattern. The environment state is defined as in [14]:

$$\mathbf{x}_{\text{obs}} = [p^{g_i}, v^{g_i}, \lambda^{g_i}, \Omega, \omega, p_{g_{i+1}}^{g_i}, \psi_{g_{i+1}}^{g_i}]^T \quad (8)$$

where the superscript g^i denotes the reference frame of the i-th gate. The position and orientation of the next gate are given by $p_{g_{i+1}}$ and $\psi_{g_{i+1}}$ respectively.

The reward function is adapted from [14][13] and encourages forward progress while penalizing high angular velocity and collisions:

$$r_k = \begin{cases} -10, & \text{if collision} \\ |p_{k-1} - p_{g_k}| - |p_k - p_{g_k}| - c|\Omega|, & \text{otherwise} \end{cases}$$

Here, the subscript k represents the current timestep. The scalar $c = 0.001$ controls the penalty on angular velocity magnitude. A collision is triggered either by ground contact or when the quadcopter exits the predefined bounding box (10 m x 10 m x 7 m). Additionally, if the drone crosses a gate plane without passing through the designated 1.5 m x 1.5 m gate (i.e. missed the gate), the episode is also terminated. This reward encourages stable, accurate, and efficient gate traversal.

http://www.imavs.org/

2.3 Leaky Integrate-and-Fire Neuron

The key element of the spiking network is the Leaky Integrate-and-Fire (LIF) neuron, a simplified yet biologically inspired model compatible with gradient-based learning frameworks, implemented using the snnTorch Python library [9]. LIF neurons accumulate input over each time step and emit discrete spikes when their membrane potential exceeds a threshold. After spiking, the LIF neuron is soft-reset, subtracting the threshold potential from the current membrane potential. The evolution of the membrane potential $U[t]$ is governed by the first-order differential equation:

$$U[t+1] = \beta U[t] + I[t] - U_{\text{th}} S[t] \tag{9}$$

$$S[t] = H(U[t] - U_{\text{th}}) \tag{10}$$

Here, $U[t]$ is the membrane potential, $\beta \in [0, 1)$ is the decay constant, $I[t]$ is the input current, U_{th} is the firing threshold, and $H(\cdot)$ denotes the Heaviside step function. The output spike $S[t] \in \{0, 1\}$, soft-resets the membrane potential by subtracting the threshold potential, denoted by the last term in Equation 9. Our implementation uses LIF neurons with a default threshold of $U_{th} = 1$ and $\beta = 0.999$. The high β minimizes the leak of membrane potential, and encourages the LIF neurons to fire earlier and more frequently. This results in a stronger gradient flow during training which promotes learning via error backpropagation [9]. This is also exploited by rate decoding the output, as explained further in subsection 2.4 along side Figure 1 and how our implementation differs from a traditional ANN.

2.4 Implementation of Spiking Neural Networks

SNNs introduce the following two key challenges when applied to gradient-based learning.

1. Spiking neurons, such as the LIF model in Equation 10, show inherent time-dependency, maintaining an internal membrane potential that spans multiple time steps. This stateful property enables temporal integration but complicates the design of standard feed-forward architectures and training methods.

2. The binary, non-differentiable nature of spike outputs, prevents the use of conventional gradient-based optimization without modification.

To address the first challenge of handling the temporal dynamics, we process only the current environment state at each forward pass, without explicitly modelling dependencies across timesteps. This design choice follows the insight of the approach by Ferede [14], where a purely feed-forward ANN was shown to achieve effective control despite the absence of a recurrent structure. This approach allows us to leverage the temporal characteristic of LIF neurons for the decoding of the output. We address the problem by using a rate encoding approach common in ANN-to-SNN conversion methods [15, 16], which we shall refer to as *cycling*. With this strategy each observation state $\mathbf{x} = \mathbf{x}_{\text{obs}}[t] \in \mathbb{R}^{20}$ at timestep t is held constant and repeatedly propagated for a fixed number of steps C (i.e. *cycles*) through the network. The network is composed of three fully connected layers of LIF neurons, parametrized by weight matrices W. During this process, each LIF neuron can accumulate membrane potential and spike across several cycles. Each observation state $\mathbf{x}$ is input using standard current injection, where the float values are directly passed to the LIF neurons with the output being binary spikes $s \in \{0, 1\}^N$, with N denoting the number of neurons in the hidden layer. A direct drawback of this cycling strategy is that we do not explicitly use the temporal modelling capabilities of SNNs, therefore operating without recurrence. Superscripts $(l) \in \{1, 2, 3\}$ indicate the corresponding network layer and at each cycle c, the spiking activations are computed as:

$$\mathbf{s}^{(1)}[c] = \text{LIF}^{(1)}(W^{(1)}\mathbf{x}, U^{(1)}[c-1]) \tag{11}$$

$$\mathbf{s}^{(2)}[c] = \text{LIF}^{(2)}(W^{(2)}\mathbf{s}^{(1)}[c], U^{(2)}[c-1]) \tag{12}$$

$$\mathbf{s}^{(3)}[c] = \text{LIF}^{(3)}(W^{(3)}\mathbf{s}^{(2)}[c], U^{(3)}[c-1]) \tag{13}$$

The output spikes from the final layer are averaged using rate decoding, as motivated in subsection 2.5, whereby the average firing rate of each neuron over the amount of cycles C is computed as shown below [17]:

$$\bar{\mathbf{S}} = \frac{1}{C} \sum_{c=1}^{C} \mathbf{s}^{(3)}[c] \tag{14}$$

The decoded spike-rate vector $\bar{\mathbf{S}} \in [0, 1]^N$, for the N neurons in the layer, is then passed to a fully connected linear output layer to produce the continuous motor commands $\hat{\mathbf{u}} \in [-1, 1]^4$ which are then mapped to actual normalized motor commands $\mathbf{u} \in [0, 1]^4$:

$$\hat{\mathbf{u}} = W_{\text{out}} \bar{\mathbf{S}} + \mathbf{b}_{\text{out}} \tag{15}$$

An important consequence of rate decoding is that, since output spikes are binary in each cycle, decoding over C cycles produces a quantized latent output.

$$s^{(c)} \in \{0, 1\}^N, \qquad \bar{S} \in \left\{0, \frac{1}{C}, \frac{2}{C}, \ldots, 1\right\}^N \tag{16}$$

The resolution is $\frac{1}{C}$ with exactly $C+1$ possible values for the average firing rate of each LIF neuron (e.g., C=1 : $\{0, 1\}$; C=2 : $\{0, 0.5, 1\}$; C=3 : $\{0, \frac{1}{3}, \frac{2}{3}, 1\}$). This discretization precedes the continuous action mapping and limits the latent output's representational resolution to $\frac{1}{C}$.

The second challenge, the non-differentiability of spikes, is caused by the discontinuous nature of the Heaviside step function $H(\cdot)$. As its derivative $\delta(\cdot) \in \{0, \infty\}$ evaluates to zero almost everywhere and diverges to infinity at the threshold, it prevents the use of exact gradient-based optimization.

http://www.imavs.org/

To solve this issue, we adopt a surrogate gradient method, wherein the true derivative of $H(\cdot)$ is replaced by a smooth, differentiable approximation during the backward pass as in [9]. Specifically, we use the derivative of a shifted arctangent function as the surrogate, adapted from [8]:

$$\frac{\delta H}{\delta U} \approx \frac{1}{\pi\left(1+(\pi U)^2\right)} \tag{17}$$

This approach preserves the ability to train the network through standard back propagation techniques while leaving the discrete spiking behaviour untouched in the forward pass.

2.5 Spiking Neural Network Architecture

The choice to specifically use spike-rate decoding is grounded in well-established findings from ANN-to-SNN conversion literature. Numerous studies have shown that under rate encoding, the firing rate of LIF neurons closely approximates the behaviour of the ReLU activation function [15]. This functional similarity has made spike-rate decoding a natural and effective choice in many prior works as it facilitates the direct transfer of architectures and training methods from ANNs to SNNs with minimal modification [18]. This similarity allows us to retain the structure of the baseline ANN used in [13], and replace the ReLU activation function in the original network with LIF neurons, as described in Figure 1. Whereas the SNN cycles each input state and uses rate decoding, while the ANN does not, the architecture of the SNN, including the number of layers and hidden units, is otherwise unchanged from the ANN which also ensures that subsequent benchmarking between the ANN and SNN is both fair and meaningful. The resulting architecture comprises three fully connected layers of LIF neurons, each containing 64 units as shown in Figure 2.

2.6 Training procedure and randomization

We train the SNN policy using the PPO algorithm [12], implemented via the Stable-Baselines3 reinforcement learning library [19]. The training setup integrates the quadcopter dynamics model described in subsection 2.1, the reward function and environment defined in subsection 2.2, and the SNN-specific adaptations introduced in subsection 2.4.

In PPO, the policy is represented by two *separate* neural networks: the actor and the value network. In our implementation, both networks share the same architecture, differing only in the dimensionality of their outputs (actor: $\in \mathbb{R}^4$, value: $\in \mathbb{R}^1$). The actor network maps a given state $\mathbf{x}$ to the parameters of a diagonal Gaussian distribution, producing a mean vector $\mu_\theta(x) \in \mathbb{R}^d$ and a log standard deviation vector $\log \sigma_\theta \in \mathbb{R}^d$, where d denotes the dimensionality of the action space (i.e. 4). Actions can then be obtained either deterministically, by taking the mean, or stochastically, by sampling from the distribution. It is important to note that for training, samples are taken stochastically:

$$a \sim \mathcal{N}\big(\mu_\theta(x), \mathrm{diag}(\sigma_\theta^2)\big), \tag{18}$$

where $a \in \mathbb{R}^4$ is the action vector and $\mathrm{diag}(\sigma_\theta^2)$ denotes a diagonal covariance matrix with entries σ_θ^2. The value network, in contrast, outputs a single scalar $V_\phi(s) \in \mathbb{R}$, which estimates the expected return from state $\mathbf{x}$. Here, θ and ϕ represent the trainable parameters of the actor and value networks, respectively.

The training procedure with PPO is accelerated by running 100 parallel environments, each simulating a single drone. This enables the agent to collect experience from multiple simulations simultaneously, significantly reducing the time required for each policy update for PPO. Episodes have a maximum duration of 12 seconds to allow the drone to fly multiple laps of the track. Simulating at a control frequency of $100\,Hz$, the 12 seconds correspond to 1200 simulation steps. We use a discount factor of $\gamma = 0.999$ to prioritize long-term rewards over immediate rewards at a given step. Lastly, a default learning rate of $\eta = 3 \times 10^{-4}$ is used. The training parameters are identical to those used to train the baseline ANN in [13], ensuring a fair and consistent comparison. To investigate the effect of the number of cycles on SNN performance, we train a separate SNN model for each cycle count in the set $\{2,\ 3,\ 4,\ 5,\ 8,\ 10\}$.

A common struggle for quadcopters is the transfer of behaviour and performance to real-life flight tests, which makes it crucial to design robust policies in simulation. To overcome this sim-to-real gap and improve robustness both in simulation and real-life, we apply domain randomization with a 30% uniform scaling on all physical parameters during training. Specifically, each parameter θ is drawn from $\theta \sim \mathcal{U}(0.7\theta_0, 1.3\theta_0)$, where θ_0 is the nominal value for the 5-inch drone identified in [13].

In addition to this, randomized initial conditions are applied at the start of each episode, enhancing the drone's ability to navigate toward the next target gate from a broader range of states. The drone's position is uniformly sampled as $x_0,\ y_0 \sim \mathcal{U}(-5,\ 5)$ m and $z_0 \sim \mathcal{U}(-3,\ 0)$ m. Linear velocities are initialized from $v_x,\ v_y,\ v_z \sim \mathcal{U}(-0.5,\ 0.5)$ m/s. Orientation is randomized over roll and pitch angles $\phi,\ \theta \sim \mathcal{U}(-\frac{\pi}{9},\ \frac{\pi}{9})$ and yaw $\psi \sim \mathcal{U}(-\pi,\ \pi)$ radians. Body angular rates are sampled as $p,\ q,\ r \sim \mathcal{U}(-0.1,\ 0.1)$ rad/s, and each motor is initialized with an angular velocity $w_i \sim \mathcal{U}(-1,\ 1)$ rad/s for $i \in \{1,\ 2,\ 3,\ 4\}$.

3 EXPERIMENTAL SETUP

We adopt the same experimental platform as in [13], using a 5-inch quadcopter configured for indoor autonomous flight. The control firmware is based on INDIflight[1], a fork of Betaflight, which runs on a STM32H743 microcontroller. State estimation is handled onboard using an Extended Kalman Filter (EKF) that fuses inertial data from a

[1] https://github.com/tudelft/indiflight

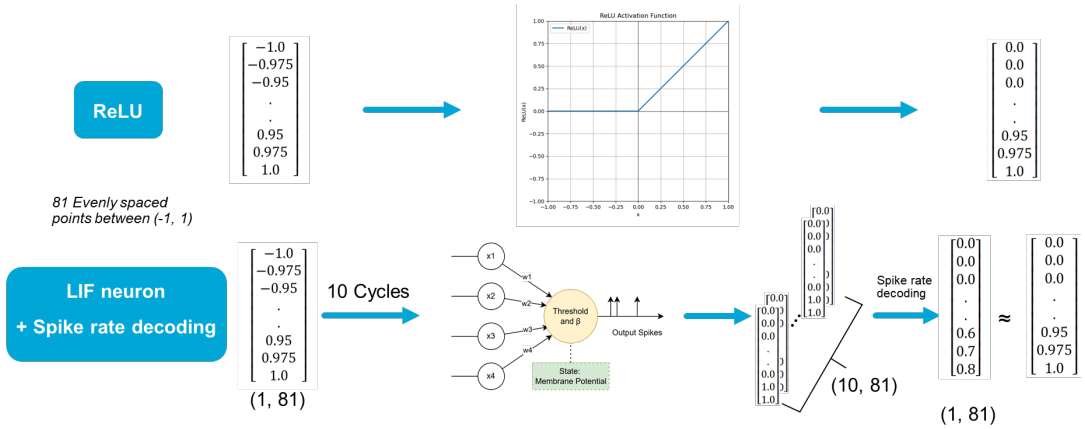


Figure 1: Comparative schematic showing how a LIF neuron with rate decoding over several cycles is similar to the ReLU activation function for an example input X of 81 evenly spaced points between [-1, 1].

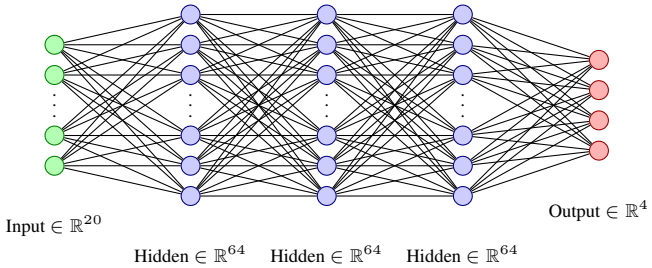


Figure 2: Schematic of the spiking neural **actor** network architecture. Each hidden layer contains 64 neurons. The value network is identical with the exception of the size of the output dimension which in contrast is 1

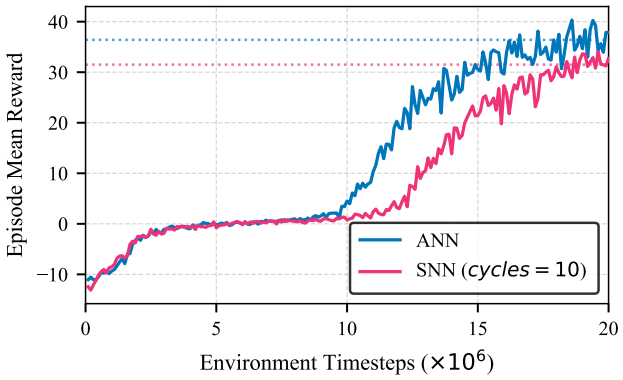


Figure 3: Training of an ANN compared to a SNN policy with 10 cycles over 20 million timesteps

TDK InvenSense ICM-42688-P IMU with external position and attitude measurements from an OptiTrack motion capture system. The state is then used as input to the SNN which outputs the corresponding motor commands. All experiments are conducted in the CyberZoo flight arena at TU Delft, a 10 m × 10 m × 7 m indoor space equipped for autonomous drone testing.

4 Results

4.1 Baseline ANN vs. SNN Performance

Using the procedure and parameters described in subsection 2.6, we train the ANN and SNN policies for a maximum of 20 million timesteps. Both models share an identical architecture, and are trained under the same conditions. Figure 3 shows the mean episode reward over the 20 million timesteps and the final converged value, taken as the average reward of the last 10% of the timesteps. The SNN model in this comparison uses a cycle count of 10, chosen on the basis of a preliminary analysis in which models were trained with cycle counts of up to 50. This analysis showed that while higher cycle counts could achieve similar or slightly improved rewards, they required substantially longer training times. A cycle count of 10 therefore provided the most favourable trade-off, achieving quite comparable reward while keeping training time manageable compared to the ANN baseline.

Each curve represents a single training run. While averaging across multiple seeds is common in RL to reduce variance [20], this was not feasible due to limited compute and long training times. All subsequent comparisons are therefore based on representative single runs.

Although the SNN demonstrates slower learning and converges to a slightly lower average reward (31.5) compared to the ANN baseline (36.4), the overall training reward convergence remains comparable. This indicates that the spiking policy is capable of effectively learning the task in simulation, with minimal degradation. However, a significant drawback of the cycled SNN becomes immediately apparent: while the ANN completed training in 16.17 minutes, the SNN required 5.98 hours to train on a standard consumer-grade lap-

http://www.imavs.org/

http://www.imavs.org/

Metric	Unsuccessful (Crash)	Successful
ANN		
Mean Reward	2.00 ± 14.52	58.23 ± 8.36
Max Reward	66.87	77.13
Crash Rate (%)	67.30	0.0
Mean Episode Length (steps)	206.0	1200.0
$\bar{v}$ (m/s)	6.39 ± 3.17	7.43 ± 2.00
v_{max} (m/s)	**28.67**	15.16
SNN (cycles=10)		
Mean Reward	-2.75 ± 10.32	$\mathbf{64.93 \pm 8.59}$
Max Reward	58.87	**87.68**
Crash Rate (%)	**44.80**	0.0
Mean Episode Length (steps)	99.2	1200.0
$\bar{v}$ (m/s)	5.56 ± 3.45	$\mathbf{7.95 \pm 1.95}$
v_{max} (m/s)	20.49	**15.43**

Table 1: **Simulation** performance comparison of ANN and SNN policies over 1000 simulated episodes. Metrics are shown for unsuccessful (crash) and successful (non-crash) episodes separately.

	ANN				SNN (cycles=10)			
Metric	T1	T2	T3	$\bar{\mathbf{T}}$	T1	T2	T3	$\bar{\mathbf{T}}$
Reward	59.38	59.93	60.00	**59.77**	70.95	70.99	69.95	**70.63**
$\bar{v}$ (m/s)	6.91	7.03	7.03	**6.99**	7.87	7.99	7.95	**7.94**
Gates	40	41	41	**40.67**	46	47	46	**46.33**

Table 2: **Real flight** performance of ANN and SNN policies over three trials.

top, with training time being proportional to the cycle count of the model.

To further compare flight performance, both models were trained with PPO until achieving a reward of $r \approx 50$ (ANN: 50.7, SNN: 50.82). Although they converged in a similar number of timesteps (ANN: 40.3×10^6, SNN: 41.1×10^6), training time differed greatly (ANN: 33 min, SNN: 10.4 h). Results over 1,000 simulated episodes are shown in Table 1.

Despite similar training rewards, the SNN achieved higher mean reward (64.93 vs. 58.23) and velocity on successful runs, with a lower crash rate (44.8% vs. 67.3%). Failures mainly resulted from extreme randomized initial conditions. The ANN tended to survive longer in failed runs but ultimately crashed more often, while the SNN either recovered or failed quickly.

The higher SNN performance is confirmed with real flight tests. Both policies were flown for three identical runs of 12 seconds each, equivalent to one fully charged battery starting from a defined hover position.

The trajectory comparison of the best performing trial of the ANN and SNN policies in Figure 4 highlights distinct behavioural differences. While the ANN follows a relatively smooth flight path with a roughly constant velocity of 7.03 m/s, The SNN policy demonstrates a more dynamic control strategy, reaching a higher average velocity of 7.99 m/s. Notably, the SNN shows a greater variation in speed, accelerating sharply through the central gate and hence decelerating more aggressively to navigate the outermost turns. This results in sharper cornering and frequent changes in acceleration. The SNN policy appears to prioritize aggressively accelerating during the straighter sections of the track, trading off smoothness for speed, resulting in the SNN outperforming the ANN in both episode reward (70.99 vs. 60.00) and gates passed (47 vs. 41).

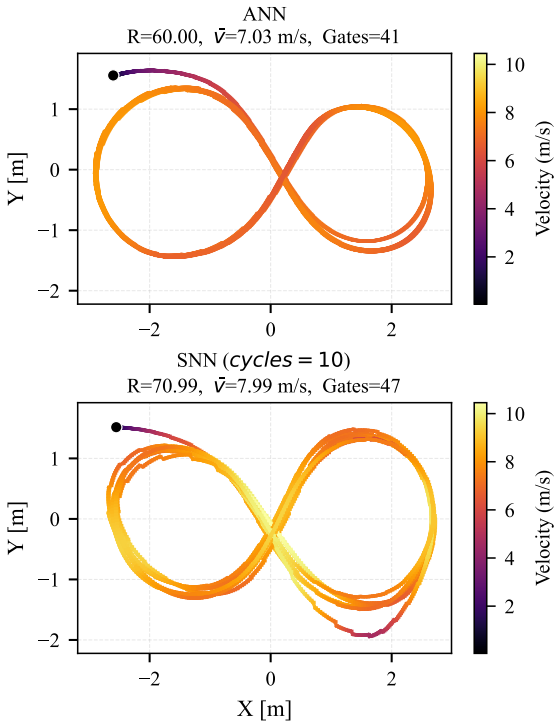


Figure 4: **Real flight** trajectory comparison between the best trials of the ANN and SNN with 10 cycles.

4.2 *Effect of Cycles on SNN Flight Performance*

To assess the effect of the cycle count on SNN training and flight performance, six models were trained using cycle counts: $\{2, 3, 4, 5, 8, 10\}$. Each model was trained for a total of 150 million timesteps, a value selected with the goal of reaching convergence in mean episode reward for all models, while maintaining feasible training time. To highlight the reward convergence, an exponential moving average (EMA) with a smoothing factor of ($\alpha = 0.1$) was applied to the mean episode reward. The converged reward is calculated as the average reward of the last 10% of the training timesteps. Figure 5 shows the evolution of the mean episode reward for the six SNN models and the converged reward

As the number of cycles increases, the converged episode

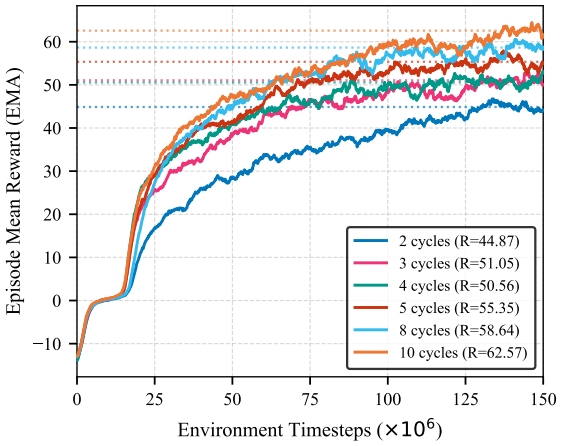


Figure 5: EMA of mean episode reward for six SNNs over 150 million timesteps.

reward improves consistently, indicating a clear performance gain from averaging the spiking output over a higher cycle count. The lowest-performing model, using only 2 cycles, converged to a mean reward of 44.87, while the best-performing model, with 10 cycles, reached a reward of 62.57. Interestingly, 3 and 4 cycles yield comparable rewards (51.05 and 50.56), while 5, 8, and 10 cycles result in increasingly higher performance (55.35, 58.64, and 62.57, respectively). This demonstrates that a higher cycle count provides finer resolution in spike-rate decoding, resulting in smoother motor commands, which in turn improves learning and performance. However, in addition to the significantly longer training time compared to the baseline ANN, a second drawback of cycled SNNs becomes evident during deployment: lower observed update frequencies. This latency is especially a challenge in real-time deployment for models with higher cycle counts.

To investigate the real-time deployment, each model was flown in three real-world trials using specific combinations of cycle count and desired control update frequency $f_{des}^{(c)}$. The goal was to compare the models under two conditions: constant computational load and constant observed control frequency. For the constant computational load condition, models and parameters were matched based on a similar inference rate τ, defined as the product of the observed control frequency and the cycle count. The average performance metrics across trials are summarized in Table 3.

Despite the hardware limitations on the observed frequency, the achieved inference rate for the combinations remains relatively stable ranging from 806 to 852 inferences per second. Table 3 shows that flight performance generally improves with increasing cycle count, peaking at cycles = 8 with the highest average reward of 70.76. The best gate completion performance, however, is achieved by the 5-cycle model,

Cycles	$f_{des}^{(c)}$ (Hz)	$f_{obs}^{(c)}$ (Hz)	τ (iter/s)	Rew.	$\bar{v}$ (m/s)	Gates
2	500	403.08	806.17	65.04	7.14	42.67
3	333	268.11	804.32	65.45	7.80	44.33
4	250	204.31	817.25	68.43	8.01	44.67
5	200	167.80	838.98	70.27	8.25	50.00
8	125	105.95	847.63	70.76	8.09	45.67
10	100	85.16	851.60	69.81	8.03	47.00

Table 3: **Real flight** average performance metrics for **inference-constrained** SNN flight tests across different cycle counts for three trials. τ denotes the amount of forward passes per second (inference rate)

Cycles	$f_{des}^{(c)}$ (Hz)	τ (iter/s)	Rew.	Progress Rew.	Rate Penalty	Gates
2	500	806.17	65.04	73.30	-8.26	42.67
3	333	804.32	65.45	74.41	-8.96	44.33
4	250	817.25	68.43	77.45	-9.02	44.67
5	200	838.98	70.27	80.66	-10.39	50.0
8	125	847.63	70.76	79.74	-8.98	45.67
10	100	851.60	69.81	79.46	-9.65	47.0

Table 4: **Real flight** average reward decomposition for **inference-constrained** tests from Table 3. Reward = Progress + Rate Penalty.

which passes 50.00 gates with a higher average velocity of 8.25 m/s and a reward of 70.27. Despite better objective performance in terms of gates passed, the lower reward of the 5 cycle model is largely attributed to the reward function's rate penalty term, as shown in Table 4, indicating the reward function can still be optimized for time-optimal trajectories.

Under similar observed frequency, results in Table 5 are consistent. The 8-cycle model gives the highest reward (70.01) with strong velocity and gates, while the 5-cycle model again leads in velocity (8.36 m/s) and gates (50.00) with competitive reward (69.81). Its observed frequency was higher than expected at 337.20 Hz, likely due to flight controller scheduling, which may have improved responsiveness.

Lower-cycle models at 333 Hz show reduced rewards and fewer gates, and the 10-cycle model benefits less from added cycles because responsiveness declines with higher computational load. Representative trajectories are shown in Figure 6.

The trajectories **(a)** and **(d)** in Figure 6 show the two highest-performing models, both achieving a reward of 71.52. Like the 10-cycle SNN in subsection 4.1, they show aggressive acceleration, sharp turns, and peak speeds on straight segments, reinforcing our earlier observation. In contrast, the 4-cycle **(b)** and 2-cycle **(e)** models achieved the lowest rewards. Interestingly, the 2-cycle model improved significantly its reward from 44.87 in simulation to 63.13 in real-

http://www.imavs.org/

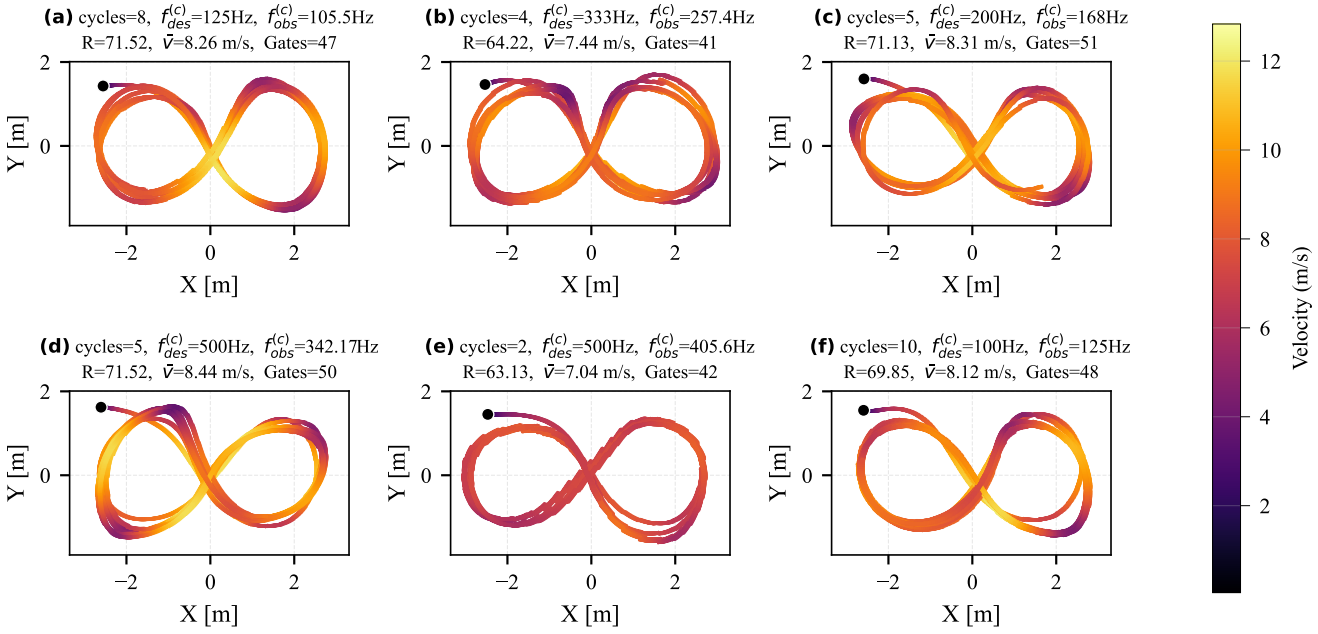


Figure 6: **Real flight** trajectories for selected trials. (a) and (d) highest reward, (b) and (e) second lowest and lowest reward, (c) highest gates passed, (f) highest cycle model. Videos: `https://drive.google.com/drive/folders/1uGINGe71wu0Hrh0_ZBDAHMIO-s6B5qd3?usp=drive_link`

Cycles	$f_{des}^{(c)}$ (Hz)	$f_{obs}^{(c)}$ (Hz)	τ (iter/s)	Rew.	$\bar{v}$ (m/s)	Gates
3	333	268.11	804.32	65.45	7.80	44.33
4	333	257.08	1028.32	64.37	7.65	41.67
5	500	337.20	1686.00	69.81	8.36	50.00
8	500	289.33	2314.67	70.01	8.30	47.67
10	500	261.07	2610.67	69.07	7.83	45.33

Table 5: **Real flight** averaged performance metrics for **frequency-constrained** SNN tests across different cycle counts for three trials.

world tests, displaying slower, smoother flight resembling the ANN baseline. Despite higher reward and velocity, the 4-cycle model passed fewer gates than the 2-cycle model. A similar mismatch appears for the 5-cycle model in trajectory **(c)**, which passed the most gates (51) but had a lower reward than the 8-cycle model, highlighting the need to refine the reward function to better reflect objective time-optimal performance.

5 Conclusion

This work presented the first successful application of a fully spiking actor-critic network trained with PPO for continuous quadcopter control. The SNN, implemented with LIF neurons and trained using surrogate gradients and spike-rate decoding, achieved performance superior to state-of-the-art ANNs in both simulated and real-world high-speed navigation tasks. Evaluation on the 5-inch racing quadcopter revealed that the SNN not only matched the ANN in control fidelity but also outperformed it in reward, robustness, and average velocity, despite its slower training and higher inference latency due to cycle-based spike integration.

An extensive analysis of different cycle counts demonstrated a clear trade-off between temporal resolution and observed update frequency. While higher cycle counts yielded smoother motor outputs and improved control performance, they also imposed greater computational costs and reduced update frequencies on embedded hardware. The 8-cycle model attained the highest average reward, whereas the 5-cycle model had the best balance between reward, velocity, and gates passed, confirming that moderate cycle counts provide the most favourable trade-off for real-time applications.

Future work will focus on extending this approach to neuromorphic hardware platforms such as Intel Loihi or other event-driven processors to fully leverage the energy efficiency of spiking neurons. To enable efficient deployment, further investigation is needed into quantizing the network weights and activations and improving runtime efficiency. Additionally, we aim to explore how varying the network parameters and architecture impacts both control performance and computational cost. These directions are critical for designing scalable, lightweight SNN controllers for embedded applications in autonomous aerial vehicles.

http://www.imavs.org/

http://www.imavs.org/

REFERENCES

[1] Robin Ferede, Guido de Croon, Christophe De Wagter, and Dario Izzo. End-to-end neural network based optimal quadcopter control. *Robotics and Autonomous Systems*, 172:104588, February 2024.

[2] Yunlong Song, Mats Steinweg, Elia Kaufmann, and Davide Scaramuzza. Autonomous Drone Racing with Deep Reinforcement Learning, August 2021. arXiv:2103.08624 [cs].

[3] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Girish Chinya, Yongqiang Cao, and et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

[4] Amirhossein Tavanaei, Mohammad Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural Networks*, 111:47–63, 2019.

[5] Carlos Zamarreño-Ramos, Luis A Camuñas-Mesa, Jorge A Pérez-Carrasco, Timothée Masquelier, Teresa Serrano-Gotarredona, and Bernabé Linares-Barranco. On spike-timing-dependent-plasticity, memristive devices, and building a self-learning visual cortex. *Frontiers in Neuroscience*, 5:26, 2011.

[6] Luca Zanatta, Francesco Barchi, Simone Manoni, Silvia Tolu, Andrea Bartolini, and Andrea Acquaviva. Exploring spiking neural networks for deep reinforcement learning in robotic tasks. *Scientific Reports*, 14(1):30648, December 2024. Publisher: Nature Publishing Group.

[7] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction, 2nd ed*. Reinforcement learning: An introduction, 2nd ed. The MIT Press, Cambridge, MA, US, 2018. Pages: xxii, 526.

[8] Wei Fang, Zhaofei Yu, Yanqi Chen, Timothee Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating Learnable Membrane Time Constant to Enhance Learning of Spiking Neural Networks, August 2021. arXiv:2007.05785 [cs].

[9] Jason K Eshraghian, Max Ward, Emre Neftci, and et al. Training spiking neural networks using lessons from deep learning. In *Proceedings of the IEEE*, volume 111, pages 1016–1054, 2023.

[10] Federico Paredes-Vallés, Jesse Hagenaars, Julien Dupeyroux, Stein Stroobants, Yingfu Xu, and Guido de Croon. Fully neuromorphic vision and control for autonomous drone flight, March 2023. arXiv:2303.08778 [cs].

[11] Stein Stroobants, Christophe de Wagter, and Guido C. H. E. De Croon. Neuromorphic Attitude Estimation and Control, November 2024. Issue: arXiv:2411.13945 arXiv:2411.13945 [cs].

[12] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017. arXiv:1707.06347 [cs].

[13] 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, April 2025. arXiv:2504.21586 [cs].

[14] Robin Ferede, Christophe De Wagter, Dario Izzo, and Guido C. H. E. de Croon. End-to-end Reinforcement Learning for Time-Optimal Quadcopter Flight. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6172–6177, May 2024. arXiv:2311.16948 [cs].

[15] Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition. *International Journal of Computer Vision*, 113(1):54–66, May 2015.

[16] Yangfan Hu, Qian Zheng, Xudong Jiang, and Gang Pan. Fast-SNN: Fast Spiking Neural Network by Converting Quantized ANN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(12):14546–14562, December 2023.

[17] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image Classification. *Frontiers in Neuroscience*, 11, December 2017. Publisher: Frontiers.

[18] Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep Residual Learning in Spiking Neural Networks, January 2022. arXiv:2102.04159 [cs].

[19] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[20] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep Reinforcement Learning at the Edge of the Statistical Precipice, January 2022. arXiv:2108.13264 [cs].

Design of Frontal Perching Mechanism to Vertical Surfaces for Flapping-Wing MAVs

Tomoyuki Matsuda, Saad Hussain, Abner Asignacion Jr., and Satoshi Suzuki*
Chiba University, Japan

http://www.imavs.org/

Abstract

In this study, a novel frontal perching mechanism that is mounted at the front of the vehicle was proposed. Inspired by the way lemurs grasp tree trunks, this design contrasts with conventional underbody mechanisms. The frontal configuration offers potential advantages in high-speed approaches and enables direct engagement with vertical targets. Load testing and flight experiments were conducted using a flapping-wing platform equipped with the proposed mechanism. The load testing revealed how the gripping performance varied depending on the surface characteristics of the object and the direction of the applied force. The results of the flight experiments demonstrated that the mechanism is capable of stable perching on vertical cylindrical structures. This study confirms the feasibility and advantages of frontal perching for FWMAVs and contributes a new perspective to perching mechanism design in aerial robotics.

1 Introduction

In recent years, the demand for drones has been increasing globally, with industrial applications such as aerial photography, mapping, exploration, and pesticide spraying becoming widespread [1]. These tasks are all conducted without physical contact between the drone and the environment. Recently, research on tasks involving environmental contact (e.g., pushing-pulling tasks, peg-in-hole tasks) has gained momentum. This study focuses specifically on the perching task among such applications. The Flapping-Wing Micro Aerial Vehicle (FWMAV), the subject of this study, excels in flight efficiency and safety but generally suffers from short flight endurance. To address this limitation, the perching task has gained attention as a promising solution.

For rotary-wing drones, multiple perching mechanisms have been proposed in prior research [2]. However, due to the small payload and design constraints of FWMAVs, existing perching mechanisms for this type of vehicle remain limited [3] [4]. In particular, weight reduction and the difficulty of precise grasping are regarded as major challenges.

This study proposes a frontal perching mechanism. Most existing perching mechanisms are mounted on the underside of the drone. Such configurations require precise alignment with the target object due to the significant reactive forces that occur upon contact between the mechanism and the surface. Moreover, when the center of mass is relatively high, these designs tend to be unstable against rotational disturbances, such as wind forces. In contrast, the proposed frontal perching mechanism is expected to enable high-speed perching and stable holding on vertical surfaces. Previous research introduced a UAV called the "Hugging-wing Robot," which perches by embracing vertical structures [5]. This study adapts that concept to FWMAVs and verifies its effectiveness. The authors of [5] have inspired the design of perching mechanisms that mimic how animals like geckos and flying squirrels leap onto trees and wrap around them. Similarly, this study adopts a bio-inspired design approach based on the wrapping behavior of arboreal animals, lemurs.

The remainder of this paper is organized as follows. In section 2, the design and development process of the frontal perching mechanism is presented. This includes the biological inspiration behind the concept, the parametric representation of the mechanism, and the associated mechanical considerations. The section also describes the manufacturing process, covering the material choices and structural configuration used to realize the proposed design. Section 3 presents the experimental evaluation used to assess the gripping performance of the proposed mechanism, including both the results and their interpretation.

2 Structure Overview

2.1 Drone Platform

Recent research on flapping-wing drones has primarily focused on two categories: ornithopters, which mimic birds, and entomopters, which mimic insects [6]. Among these, entomopters are particularly promising for perching applications due to their high agility and precise maneuverability during flight. Integrating a perching manipulator into such platforms is expected to significantly enhance their mission capability.

In this study, the Entomopter Nimble+ by Flapper Drones [7] was used as the experimental platform, owing to its VTOL capability and agile 6-DoF maneuverability, both of which are well suited for mid-air perching experiments. Figure 1 shows a photograph of the Nimble+, along with the motion capture deck and battery used in the experiment. The vehicle is based on the DelFly Nimble [8], a well-established entomopter platform. These features facilitate mid-air perch-

*Email address: suzuki-s@chiba-u.jp

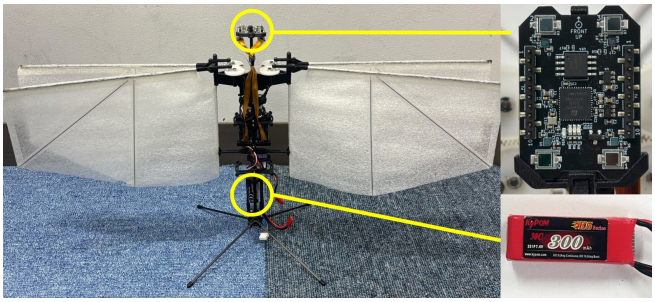


Figure 1: Entomopter "Nimble+", Lighthouse and Battery

ing experiments. The basic specifications of the Nimble+ are listed in Table 1. Like most FWMAVs, its payload capacity is limited, with 25 g recommended. However, flight is possible with up to 50 g under experimental conditions, albeit with reduced stability. This constraint presents a critical challenge for integrating perching mechanisms. Accordingly, the frontal perching mechanism was designed to be compatible with the Nimble+ and manufacturable within its payload limit.

2.2 *Design and Parameters*

To design the frontal perching mechanism, inspiration was drawn from the locomotion strategies of arboreal mammals that climb vertical trees. Incorporating biological insights from tree-dwelling animals into flying robot design is valuable due to their adaptive and efficient behaviors. Figure 2 compares the gripping posture and forces of a lemur grasping a vertical object with those of the proposed mechanism. As shown in the left panel of Figure 2, the lemur supports its body using two limbs above and two below the object. This limb configuration served as a reference for the design of the mechanism. Furthermore, [9] provides a detailed analysis of the weight distribution between the upper and lower limbs, highlighting how lemurs effectively support their body weight against vertical objects.

The detailed design of the frontal perching mechanism is presented in two orthographic views: the *Side View* (Figure 3) and the *Top View* (Figure 4). All performance-critical parameters are annotated in the figures and described below.

In the side view, the mechanism includes an additional

Parameters	Values
Weight (Without Shells)	102 g
Payload (Recommended)	25 g
Height	25 cm
Wingspan	49 cm
Battery	LiPo 2S 300 mAh
Flight time (with 25 g payload)	5 min

Table 1: Specifications of the Entomopter "Nimble+"

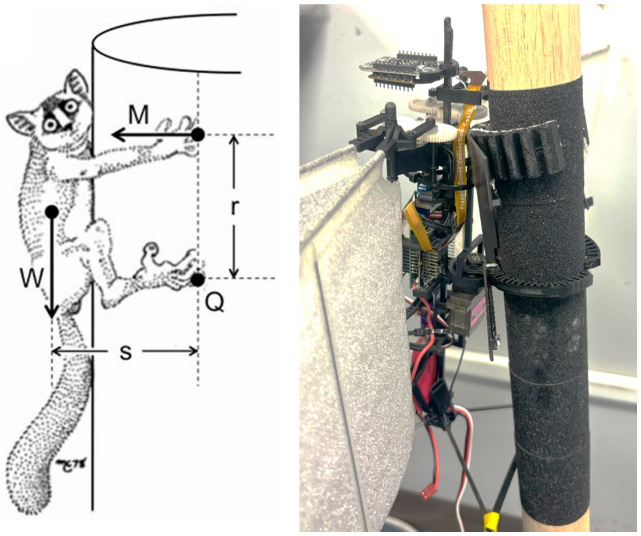


Figure 2: Lemur Grasping a Vertical Object [9] and the Proposed Frontal Perching Mechanism

non-gripping stopper located beneath the lower fingers, in contrast to the lemur-inspired configuration, which employs only four gripping limbs. This stopper is designed to resist the pitching moment generated by the vehicle. The widths of the upper and lower fingers are denoted by $t_1 = 20$ mm and $t_2 = 5$ mm, respectively, while the vertical distance between the upper and lower fingers is represented by $d_1 = 40.2$ mm and the distance between the lower fingers and the stopper is represented by $d_2 = 125$ mm. The lower fingers are arranged in a staggered parallel configuration, with an offset of 5 mm introduced between them. This design prevents interference between the fingers by maintaining parallel alignment [10]. The entire mechanism is mounted with a forward tilt of $\theta = 11.4$ deg. This inclination allows the moment induced by the weight of the vehicle to be partially supported by the stopper, reducing finger load. It also improves alignment with vertical objects during forward flight, particularly when the vehicle approaches the target in a nose-down posture typical of high-speed perching. In the top view, the upper and lower fingers appear to overlap but differ in length, denoted by $l_1 = 64.8$ mm and $l_2 = 82.2$ mm, respectively. Each finger is inclined at an angle $\phi = 30$ deg, and the lateral spacing between the left and right finger pairs is defined as $a = 11$ mm. The distance between the mechanism and the airframe, denoted as $b = 16$ mm, arises from the structural design but is ideally minimized for compact integration.

No formal parameter optimization was conducted in this study. Instead, the values were heuristically chosen to match a cylindrical object with a diameter of $D = 40$ mm. Due to the high dimensionality of the parameter space, an exhaustive search for optimal values was deemed impractical at this stage. Future work will explore numerical simulations or an-

http://www.imavs.org/

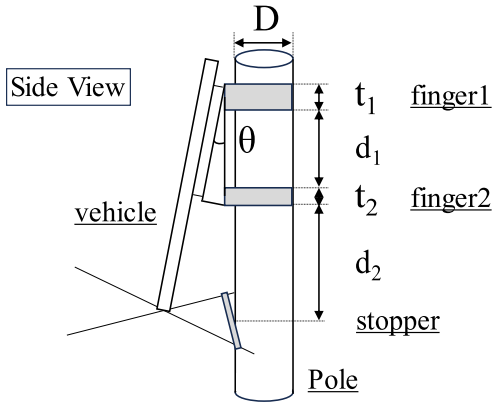


Figure 3: Side View of the Design

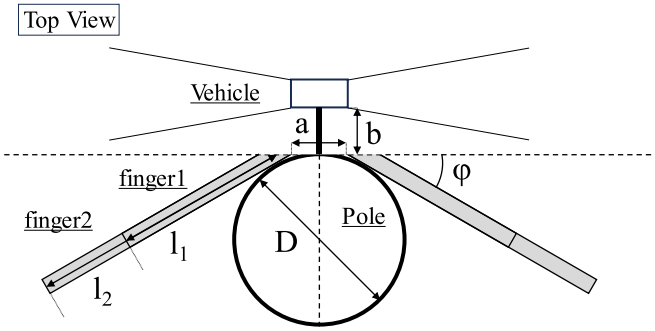


Figure 4: Top View of the Design

alytical modeling to refine the design.

The side view shown in Figure 5 illustrates the directions of the reaction forces acting on each component, as well as the positions and distances of the corresponding contact points. $c = 124$ mm represents the distance from the center of mass to the contact point of the stopper and $\alpha = 19$ deg represents the angle the stopper makes with the horizontal. Among these, F_1 and F_2 represent the reaction forces generated by the upper and lower fingers, respectively. Strictly speaking, these forces are not unidirectional as depicted. According to [9], lemurs primarily use their upper limbs to support vertical loads, while their lower limbs resist shear forces. Based on this observation, the fingers were asymmetrically designed—thicker and shorter at the top, thinner and longer at the bottom. However, in this mechanism, the center of mass is located relatively high, and therefore, the roles of the upper and lower fingers were reversed to better accommodate the shifted load distribution. This configuration is intended to efficiently distribute the moment generated by the weight of the vehicle across all contact points and enhance overall perching performance.

A simplified static analysis was conducted using the presented parameters to evaluate the equilibrium of the system. The equilibrium conditions in the horizontal and vertical directions, as well as the moment balance around the contact point of the stopper, are expressed in equations (1)–(3).

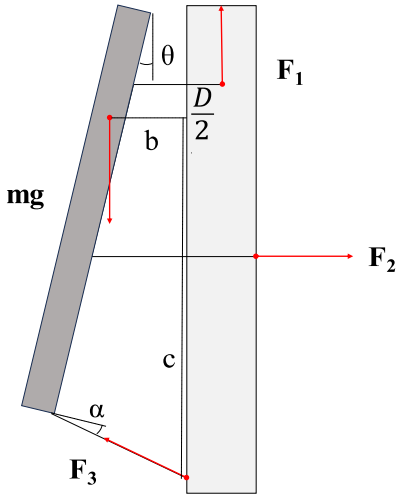


Figure 5: Force Diagram of the Side View

$$F_2 = F_3 \cos(\theta + \alpha) \tag{1}$$

$$F_1 + F_3 \sin(\theta + \alpha) = mg \tag{2}$$

$$F_1 \cdot \frac{D}{2} + mg \cdot b - F_2 \cdot d_2 = 0 \tag{3}$$

By solving these simultaneous equations, the values of F_1, F_2, and F_3 can be obtained as follows.

$$F_1 = \frac{2mg(d_2 - b\tan(\theta + \alpha))}{2d_2 + D\tan(\theta + \alpha)} \tag{4}$$

$$F_2 = \frac{mg(D + 2b)}{D\tan(\theta + \alpha) + 2d_2} \tag{5}$$

$$F_3 = \frac{mg(D + 2b)}{D\sin(\theta + \alpha) + 2d_2\cos(\theta + \alpha)} \tag{6}$$

Substituting the known values into the equations yields $F_1 = 1.16$ N, $F_2 = 0.30$ N, and $F_3 = 0.35$ N, respectively. These results indicate the load applied at each contact point, representing the weight distribution. In this case, the result shows that a particularly large load is applied to F_1.

It should be noted that these calculated forces represent the minimum values required to maintain static equilibrium. Among them, F_3 corresponds to a passive reaction force generated at the stopper contact point. In practical operation, if the actual gripping forces provided by F_1 and F_2 exceed their respective calculated values, the mechanism is expected to be capable of maintaining stability, even under small variations in external loading or contact conditions.

http://www.imavs.org/

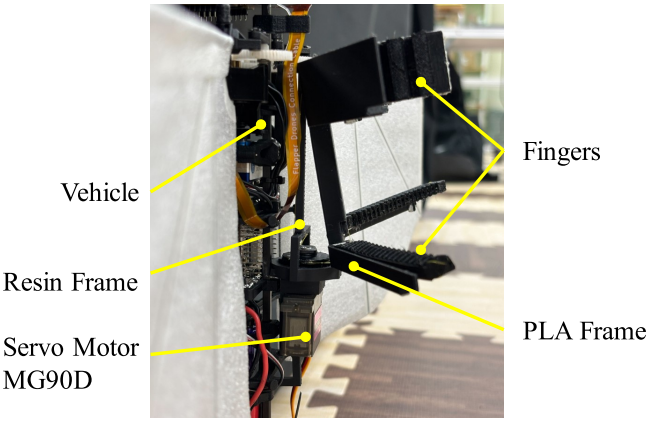


Figure 6: Attached Frontal Perching Mechanism

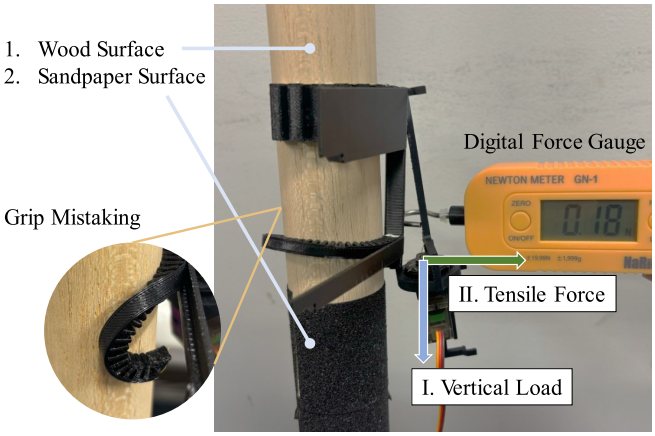


Figure 7: Experimental Setup for Load Testing

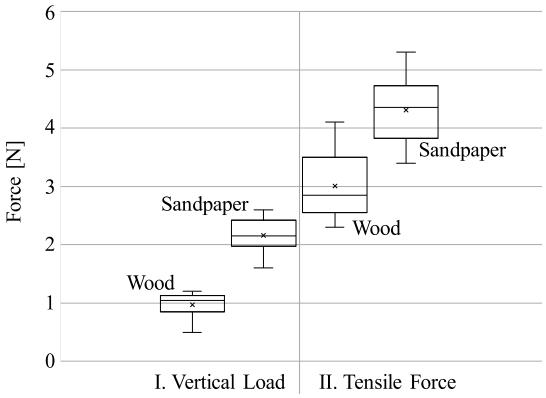


Figure 8: Load Testing Results

2.3 *Manufacture*

Based on the designed structure and parameters, the frontal perching mechanism was manufactured. The mechanism comprises four sections: the servo unit, resin frame, PLA frame, and finger components. Each part was manufactured separately and bonded together using adhesive.

First, the servo unit utilizes an MG90D micro servo motor. This model was selected due to its internal metal gears, which offer increased durability under the high torque loads experienced during gripping. A pulley is mounted on the servo to wind the tendon, thereby closing the fingers. Because the upper and lower fingers require different winding lengths, the pulley diameters were adjusted accordingly.

Next, the resin frame connects the mechanism to the vehicle and thus must withstand significant structural loads. To meet this requirement, the frame was printed using a stereolithography (SLA) 3D printer, which offers high material strength. Due to the angled mounting of the mechanism, the frame has a relatively complex geometry. Although SLA-printed components are generally heavier than fused deposition modeling (FDM) counterparts due to added support material, they are better suited for intricate, single-piece designs.

The PLA frame was printed using an FDM printer with hollow infill to reduce weight. It is combined with the resin frame to secure the fingers and to function as a rear support structure that absorbs impact.

The finger components were fabricated using TPU filament with an FDM printer. This soft and elastic material allows the fingers to return to their original shape when the tendon is released. They grip by tendon tension and reset by their elasticity. To enhance friction, the upper fingers were coated with silicone (Ecloflex 00-30).

All parts were bonded with adhesive and the assembled mechanism was mounted onto the vehicle using the resin frame. The adhesive used (LOCTITE 401) exhibits a tensile strength exceeding 5 MPa on resin materials, ensuring that the parts are highly unlikely to separate during operation.

The completed setup is shown in Figure 6. The total weight of the mechanism is 35 g. A PLA rod is attached to the leg of the vehicle to act as the lower stopper. As shown, the four upper fingers perform the gripping, while the stopper supports the pitching moment induced by the weight of the vehicle.

3 Experiments

To evaluate the performance of the mechanism, load testing and flight experiments were conducted.

3.1 *Load Testing*

This experiment was conducted to evaluate the performance of the fingers under lateral and vertical loading conditions. Load testing was conducted without attaching the mechanism to the vehicle. The experimental setup is shown in Figure 7. First, the mechanism was made to grip onto either a wooden surface or a sandpaper surface. The sandpaper was bonded to the wooden stick using adhesive (LOCTITE 401). If a grip mistake occurred, as illustrated in the figure, the gripping procedure was repeated. A digital force gauge was hooked onto the mechanism and used to apply force in two directions: (I) vertically downward and (II) horizontally

http://www.imavs.org/

http://www.imavs.org/

(tensile direction). As the mechanism was pulled, the gauge reading increased until the fingers lost grip. The force at the point of release, in Newtons, was recorded by visual estimation. Each condition was tested ten times, and the results were plotted as box plots in Figure 8. Because the purpose of this test was to investigate the effects of load direction and surface material on gripping performance, factors such as pulling speed and visual estimation error were not considered.

The results shown in Figure 8 indicate that the mechanism exhibits weaker performance under vertical loads but demonstrates stronger resistance to tensile forces. This trend was consistent across both wooden and sandpaper surfaces. Additionally, greater static friction on the gripping surface was found to correspond to higher load resistance.

These findings suggest that the mechanism's vulnerability to vertical loading arises from the torque applied to the upper fingers when downward forces are exerted, causing grip failure. This interpretation is supported by the mechanical analysis in Section 2.2, which indicates that the upper finger (associated with F_1) bears the largest share of the total load. To improve performance under vertical loading conditions, it may be beneficial to either enhance the gripping strength of the upper fingers or revise the finger arrangement to reduce the load concentrated on F_1. In contrast, the strong resistance to tensile forces can be attributed to the lower fingers wrapping around the object and generating substantial opposing forces.

3.2 *Flight Experiments*

To further validate the mechanism, flight experiments were conducted to evaluate its ability to approach and perch on vertical surfaces during flight. The vehicle was tracked using a lighthouse positioning system, and command values were sent to guide it toward a target object. Although the Nimble+ deck is capable of integrating servo motors for perching, a separate input system was used in this experiment: gripping and release commands were issued from a laptop. To avoid damage, high-speed collision and grasping tests were not performed. Instead, the vehicle hovered near the object before executing the gripping and stopping sequence.

Figure 9 shows sequential images of the experiment. The vehicle successfully completed the sequence of takeoff, approach, hovering, gripping, and perching. While grasping failed on wooden surfaces, successful perching was achieved on sandpaper, which offers higher surface friction.

The experimental results are plotted in Figure 10. The graphs show the reference position (ref) and the actual trajectory under both PID and robust (Non-linear disturbance observer-based) control [11]. With simple PID control, tracking performance was insufficient. However, by employing a robust controller, tracking accuracy was significantly improved. This improvement coincided with the front-mounted mechanism shifting the center of mass forward and increasing total weight to 35 g, which exceeds the recommended payload for the platform.



Figure 9: Sequential Images of the Flight Experiment from Takeoff to Perching

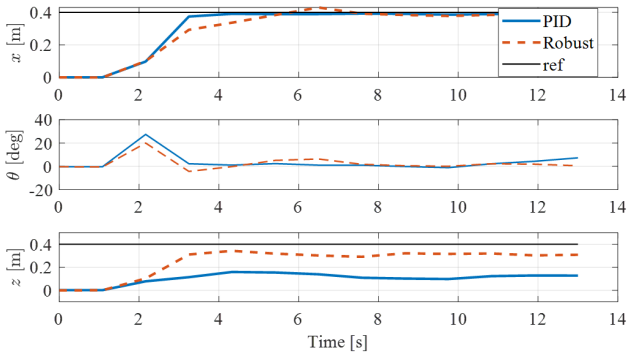


Figure 10: Time Series of Position, Pitch Angle, and Altitude during the Flight Experiment

These results confirm that the proposed frontal perching mechanism is capable of achieving stable perching during flight. Compared to underbody perching, which typically requires precise positional control due to sensitivity to rotational disturbances, the frontal configuration proved to be more tolerant of minor alignment errors and allowed easier re-takeoff.

However, achieving stable approach and gripping required the implementation of a robust controller, thereby offsetting one of the main advantages of using an entomopter—namely, its capacity for precise and responsive flight control.

This limitation is likely caused by the increased weight and the forward shift in the center of gravity due to the mounted mechanism. Therefore, further improvements such as weight reduction and mass distribution optimization could enable effective flight even under simpler PID control, restoring the full benefits of entomopter dynamics.

4 Conclusion and Future Work

In this study, we developed a novel manipulator referred to as the frontal perching mechanism for FWMAVs and successfully validated its effectiveness. By integrating a perching structure inspired by arboreal animals—capable of wrapping around vertical objects—into bird-inspired flying robots, this work opens up new avenues for perching mechanisms. The proposed mechanism combines the method of supporting vertical loads through friction (upper fingers) with the method of drawing the vehicle toward the object using strong tensile forces, enabling efficient perching on vertical structures compared to simpler claw-based or adhesive-based perching approaches. A simplified static analysis was also conducted to examine the forces acting on VTOL-type FWMAVs, identifying key design parameters critical for effective gripping performance.

For future work, the primary focus will be on reducing the weight of the system to enable stable flight using simpler controllers. More advanced flight experiments will be conducted to analyze the effects of approach speed, object shape, and surface properties on perching success. Additionally, since the current design parameters were determined heuristically, future efforts will include stress analysis and parameter optimization to enhance overall performance.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number JP23K13274 (Grant-in-Aid for Early-Career Scientists)

References

[1] A. Ollero, M. Tognon, A. Suarez, D. Lee, and A. Franchi. Past, present, and future of aerial robotic manipulators. *IEEE Transactions on Robotics*, 38(1):626–645, 2022.

[2] W. R. T. Roderick, M. R. Cutkosky, and D. Lentink. Bird-inspired dynamic grasping and perching in arboreal environments. *Science Robotics*, 6(61), 2021.

[3] Krispin C. V. Broers and Sophie F. Armanini. Repeatable energy-efficient perching for flapping-wing robots using soft grippers, 2024. arXiv preprint arXiv:2409.11921.

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

[5] Mohammad Askari, Michele Benciolini, Hoang-Vu Phan, William Stewart, Auke J. Ijspeert, and Dario Floreano. Crash-perching on vertical poles with a hugging-wing robot. *Nature Communications Engineering*, 3(98), 2024.

[6] Rafee Nekoo S, Rashad R, De Wagter C, and et al. A review on flapping-wing robots: Recent progress and challenges. *The International Journal of Robotics Research*, 2025.

[7] Flapper Drones. Flapper drones - bioinspired flying robots. *https://flapper-drones.eu/nimbleplus/*. July 9th, 2025.

[8] Matěj Karásek, Florian T. Muijres, Christophe De Wagter, Bart D. W. Remes, and Guido 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.

[9] Laura E. Johnson, Jandy Hanna, and Daniel Schmitt. Single-limb force data for two lemur species while vertically clinging. *American Journal of Physical Anthropology*, 158(3):463–474, 2015.

[10] Askari Mohammad, Shin Won Dong, Lenherr Damian, Stewart William, and Floreano Dario. Avian-inspired claws enable robot perching or walking. *IEEE/ASME Transactions on Mechatronics*, 29(3):1856–1866, 2024.

[11] A. Asignacion, S. Hussain, T. Matsuda, and S. Suzuki. Nonlinear disturbance observer-based robust control for flapping-wing aerial manipulators. *submitted to IEEE Trans of Mechatronics*, 2025. Under review.

http://www.imavs.org/

Pecking Mechanism inspired by Avian Cranial Kinesis for Flapping Wing Aerial Manipulators

Saad Hussain, Abner Asignacion Jr. and Satoshi Suzuki*
Chiba University, Japan

http://www.imavs.org/

Abstract

This study introduces a new manipulation system for Flapping Wing MAVs (FW-MAVs) based on avian beak movements. The design draws inspiration from avian Cranial Kinesis—the bone movement mechanism in bird skulls—to create a lightweight grasping/pecking mechanism for FW-MAVs. In bird anatomy, coordinated muscle actions drive skull bone movements, enabling complex tasks like pecking and grasping. The proposed design simplifies this biological system into a single-degree-of-freedom mechanism suitable for lightweight FW-MAVs. Key features of bird skull anatomy are adapted through simplified bone structures and a flexible neck to replicate natural movements. The resulting mechanism achieves wide beak opening and strong gripping force while maintaining a minimal weight of 15.3 g. Static and flight tests evaluated system performance across multiple tasks: object transport, leaf manipulation, plant sample collection, and suspended target interaction. All tasks were successfully completed with maintained flight stability.

1 Introduction

The field of micro air vehicles (MAVs) has experienced significant expansion, with applications emerging in surveillance, search and rescue operations, and structural inspections. Conventional MAVs utilize high-speed propellers, which present safety concerns in close-quarter operations. An alternative approach based on flapping wing MAVs (FW-MAVs) addresses these limitations through bio-inspired wing movements. The resulting systems exhibit reduced noise levels and increased efficiency, particularly in confined spaces and sensitive environments. Despite these advantages and inherent mechanical resilience, the challenge of enabling physical environmental interaction in FW-MAVs remains to be addressed.

To address this challenge and leverage these characteristics, researchers have developed bio-inspired perching systems based on avian and insect behavior[1, 2, 3]. Such systems enable FW-MAVs to attach to various surfaces and objects, facilitating inspection and data collection tasks while minimizing energy consumption compared to sustained hovering. However, current designs are limited to perching functionality and lack capabilities for more complex interactions such as pecking, grasping, pulling, or pushing. Despite the potential advantages FW-MAVs hold over conventional MAVs for sample collection and manipulation tasks, research on systems enabling these advanced interactions remains limited and largely impractical.

Consider, for instance, existing dual-arm manipulators designed for ornithopters [4, 5, 6]. These systems combine six servo motors and two scissor-equipped arms, but their 100 g weight restricts their use to only large-scale FW-UAVs. Their rigid design also makes quick, reliable movements challenging. Similarly, while perching grippers for conventional multirotor UAVs [7] can effectively grasp and pull objects, they lack the ability to perform pushing or pecking actions. Therefore, a new approach to interact with the environment is necessary to enable advanced aerial manipulation with FW-MAVs.

Avian beak systems provide remarkable environmental interaction capabilities through specialized mechanisms. The integration of beak structures with flexible neck movements enables tasks ranging from precise manipulation to forceful interactions. Species-specific adaptations can be observed: falcons and hummingbirds demonstrate aerial precision in prey capture and nectar feeding, while woodpeckers and sparrows exhibit powerful pecking abilities in foraging and nest construction. These functions arise from two key anatomical features: the flexible neck system and a specialized bone movement mechanism known as Cranial Kinesis.

The skull movement mechanism operates through coordinated jaw muscle activation for bone movement control. This biological system achieves operational efficiency through a combination of wide beak opening and precise positional control, facilitating both forceful and delicate object manipulation.

Analysis of avian Cranial Kinesis biomechanics reveals key principles in bone configuration, joint structure, and muscle arrangement, enabling the development of lightweight and effective mechanisms for aerial manipulation in FW-MAVs. Although the avian skull comprises multiple muscles and bones, prior studies [8, 9] have demonstrated the possibility of expressing essential beak movement as a single degree of freedom (1-DoF) mechanism. Such simplification provides an ideal model for ultra-light manipulator design in lightweight FW-MAVs. The integration of this bio-inspired

*Email address: suzuki-s@chiba-u.jp

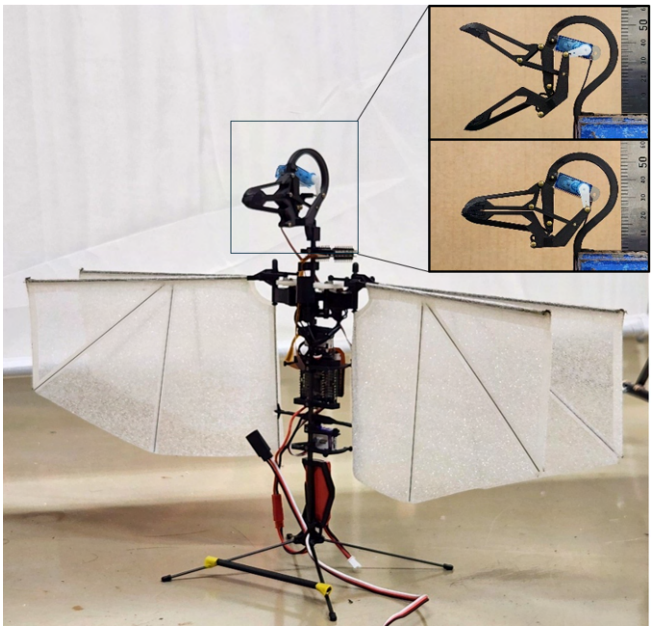


Figure 1: Pecking mechanism mounted on the Nimble+

mechanism with existing perching systems enables advanced manipulation capabilities such as pecking, grasping, pulling and pushing, thus expanding FW-MAV functionality beyond basic environmental interaction.

These biomechanical insights informed the development of a novel pecking mechanism that integrates avian Cranial Kinesis principles with a compliant neck design, replicating essential beak movements while providing impact resilience. Section 2 presents the mechanism details, followed by performance evaluation tests in Section 3. The paper concludes with Section 4.

2 Design

Drawing from avian Cranial Kinesis principles, this study presents a pecking mechanism with kinematic architecture analogous to the avian skull structure, incorporating a passively compliant neck for impact absorption. The experimental platform utilizes the Nimble+ system from *Flapper Drones*, an X-shaped FW-MAV selected for its suitable configuration. Table 1 shows the specifications of the Nimble+ platform, while Figure 1 illustrates the integration of the mechanism with the airframe. A detailed analysis of the mechanism's architecture and operation follows in Section 2.2.

2.1 Avian Cranial Kinesis

To create an effective and lightweight pecking mechanism, we began by studying how birds move their beaks. Research has identified three main types of skull movement in vertebrates with hinged upper jaws [8, 9]. These types are named based on where the hinge connects the upper beak to the skull: metakinesis (hinge at the skull's rear), mesokinesis

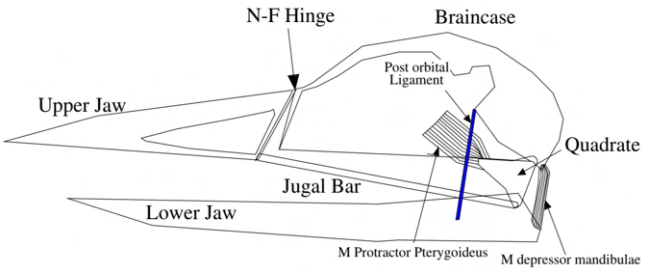


Figure 2: The mechanism of Avian Cranial Kinesis [9]

(hinge between the forehead and crown bones), and prokinesis (hinge between the nose and forehead bones). Birds typically use prokinetic skull movement, which allows them to open their beaks extremely wide. This wide opening is crucial for feeding, as it creates a large target area and makes it easier to catch or grasp objects.

As described in [9] and shown in Figure 2, a bird's skull uses five key bones to move its beak: (1) the upper jaw, (2) a connecting rod called the jugal bar, (3) the lower jaw, (4) a pivot bone called the quadrate, and (5) the braincase. The braincase stays fixed and connects to the upper jaw through a flexible hinge called the nasal-frontal (N-F) hinge. When muscles move the quadrate bone back and forth, the jugal bar transfers this motion to raise and lower the upper jaw. This efficient design works because muscles only need to move the quadrate and lower jaw, while the braincase provides a stable anchor point.

In prokinesis, the most common form of Cranial Kinesis in birds, forward movement of the quadrate bone initiates a coordinated motion sequence: the jugal bar is displaced anteriorly, elevating the upper jaw, while simultaneously the lower jaw rotates ventrally around the post-orbital ligament's terminus, creating a wide gape. The reverse motion occurs when the quadrate shifts posteriorly—the jugal bar draws the upper jaw downward while the lower jaw executes a clockwise rotation about the same pivot point, closing the beak.

This biomechanical system demonstrates remarkable efficiency: despite the involvement of multiple muscles in avian Cranial Kinesis, the fundamental jaw movement can be reduced to the singular motion of the quadrate bone. This mechanical simplification enables the entire system to be mod-

Weight (Without Shell)	102 g
Payload	25 g
Height	25 cm
Wingspan	49 cm
Battery	LiPo 2S 300 mAh
Flight time (with 25 g payload)	5 min

Table 1: Specifications of Nimble+ by *Flapper Drones*

http://www.imavs.org/

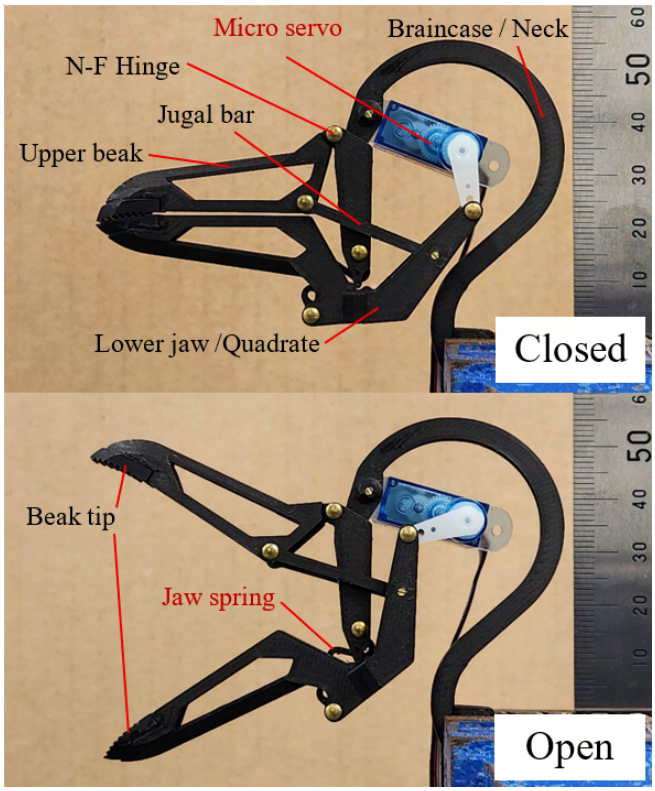


Figure 3: The proposed grasping/pecking mechanism

eled as a single-degree-of-freedom (1-DoF) mechanism.

2.2 *Mechanism*

As outlined in Section 2.1, birds achieve beak movement through a system where muscles connect to two key components: the quadrate bone and lower jaw. The quadrate serves as the primary driver, rotating around a fixed point in the skull to create the desired motion.

Our study of beak movements in common birds revealed two key observations:

- The quadrate bone and lower jaw rotate around points that are very close together
- The post orbital ligament shows minimal pivoting motion

These anatomical characteristics enabled significant design simplification. The post orbital ligament functions effectively as an extension of the fixed braincase structure, allowing the integration of the quadrate and lower jaw into a single L-shaped component. This unified structure rotates around the original post orbital ligament connection point at the braincase tip.

Based on these simplifications, the avian Cranial Kinesis mechanism can be reduced to fewer components while maintaining functional equivalence. Applying these principles, a lightweight grasping/pecking mechanism was developed that replicates the essential kinematic behavior of avian Cranial Kinesis with minimal structural complexity. The implemented design is illustrated in Figure 3.

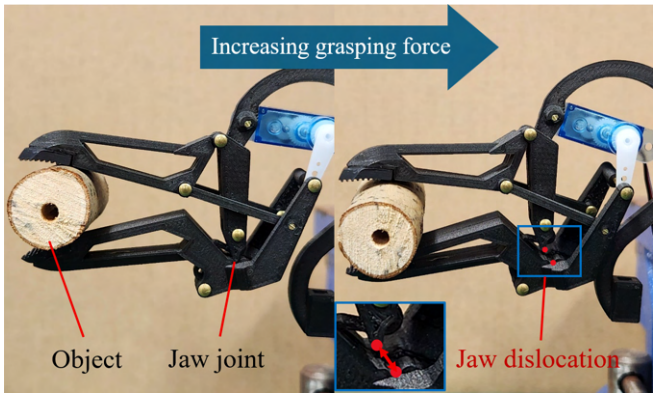


Figure 4: Dislocation of the jaw joint

The proposed design consists of five key components: the upper beak, a combined lower jaw-quadrate element, a motion-transferring jugal bar, a braincase-neck structure, and a servo motor system. Jaw movement is driven by a compact 4.9 g servo motor, including wiring, through rotation at the jaw joint. The resulting rotation is transferred through the jugal bar to control upper beak position. Grooved surfaces are incorporated into the beak tips to enhance grip strength during object manipulation.

The implemented 1-DoF mechanism achieves a minimal mass of 15.3 g and is designed for dorsal mounting on the FW-MAV platform, analogous to avian head positioning. The integration configuration is illustrated in Figure 1.

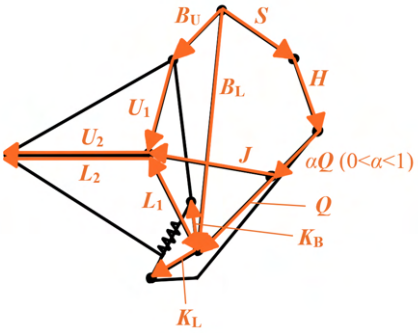
Complementing the simplified kinematic structure, the mechanism incorporates two key features for enhanced operational robustness: a compliant neck interface and an impact-mitigation system. The compliant neck structure provides a flexible coupling between the mechanism and the FW-MAV airframe, facilitating impact absorption during environmental interactions. Additionally, the mechanism features a spring-loaded dislocatable jaw joint, engineered to provide controlled mechanical compliance during high-force interactions.

During high-force interactions, the design incorporates controlled jaw dislocation. This feature allows the integrated spring to absorb excess energy and modulate the applied force, thereby protecting the structural integrity of the system. Figure 4 illustrates this adaptive behavior during the grasping of a round object.

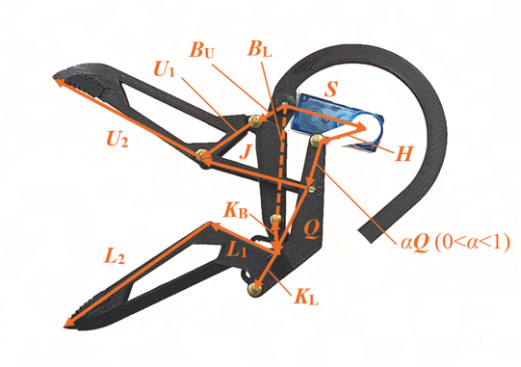
2.3 *Link Model*

A simplified link model was developed to mathematically describe the mechanism's kinematics. Figure 5 presents the link-based structure, while Figure 6 illustrates the corresponding joint configurations, including initial angles (a) and variable angles (b). The mathematical formulation enables

http://www.imavs.org/

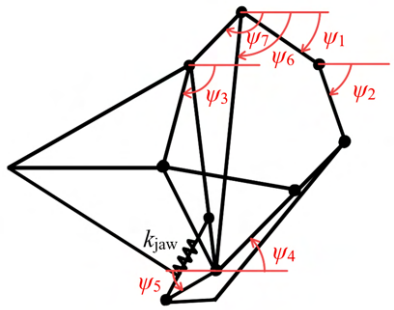


(a) Vector parameters of the link model

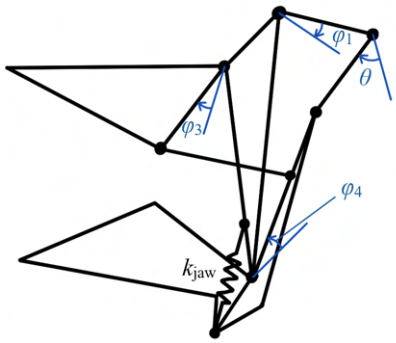


(b) Vector parameters showed on the mechanism

Figure 5: Link model: vector parameters



(a) Initial angles



(b) Variable angles

Figure 6: Link model: angle parameters

prediction of beak opening width and operational angles.

The norms of the vectors in Figure 5, representing the physical link lengths, are denoted by the same letters as their corresponding vectors (e.g., S represents the magnitude of vector $\boldsymbol{S}$). In Figure 6, angles are denoted by ψ and ϕ, where ψ represents the initial fixed angles and ϕ denotes the variable joint angles during operation. The input angle θ corresponds to the servo motor rotation angle.

The mechanism's operation begins with the rotation of input vector $\boldsymbol{H}$ about the terminus of vector $\boldsymbol{S}$. This rotation actuates the quadrate portion of the lower jaw $\boldsymbol{Q}$, inducing rotation about the jaw joint. Simultaneously, this motion drives the jugal bar $\boldsymbol{J}$, which actuates the upper beak $\boldsymbol{U_2}$ through rotation about the N-F hinge $\boldsymbol{B_U}$, resulting in jaw opening. During this sequence, the jaw spring link $\boldsymbol{K_L}$ undergoes rotation, extending the jaw spring k_{jaw} and storing potential energy for subsequent grasping operations.

Now that the link model is defined, mathematical relationships describing the beak movements can be developed. These equations will help determining the beak's opening width and angle during operation. For consistency in following calculations, the $\boldsymbol{S}$ is set as a reference point (origin).

Using the upper beak vector $\boldsymbol{U_2}$ and the lower jaw vector $\boldsymbol{L_2}$, the beak tip position vectors can be expressed as:

$$\boldsymbol{U} = \boldsymbol{B_U} + \boldsymbol{U_1} + \boldsymbol{U_2} \tag{1}$$

$$\boldsymbol{L} = \boldsymbol{B_L} + \boldsymbol{L_1} + \boldsymbol{L_2} \tag{2}$$

Here, $\boldsymbol{U}$ and $\boldsymbol{L}$ are the upper and lower beak tip position vectors, respectively. A detailed formulation can be obtained from the lengths and angles of the links.

$$\begin{bmatrix} U_x \\ U_y \end{bmatrix} = \begin{bmatrix} -B_U \cos\psi_7 + U_1 \cos(\psi_3 + \phi_3) - U_2 \cos\phi_3 \\ B_U \sin\psi_7 + U_1 \sin(\psi_3 + \phi_3) - U_2 \sin\phi_3 \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} L_x \\ L_y \end{bmatrix} = \begin{bmatrix} -B_L \cos\psi_6 + L_1 \cos(\psi_4 + \phi_4) - L_2 \cos\phi_4 \\ B_L \sin\psi_6 + L_1 \sin(\psi_4 + \phi_4) - L_2 \sin\phi_4 \end{bmatrix} \tag{4}$$

Beak tip opeining distance is given by the vertical component of the difference between the upper and lower beak tip position vectors. By substituting equation 3 and equation 4, the beak tip opening distance d_{tip} can be expressed as:

$$\begin{aligned} d_{tip} &= |\boldsymbol{U} - \boldsymbol{L}|_y \\ &= B_U \sin\psi_7 + U_1 \sin(\psi_3 + \phi_3) - U_2 \sin\phi_3 \\ &\quad - [B_L \sin\psi_6 + L_1 \sin(\psi_4 + \phi_4) - L_2 \sin\phi_4] \end{aligned} \tag{5}$$

The same can be done for the beak root opening distance

http://www.imavs.org/

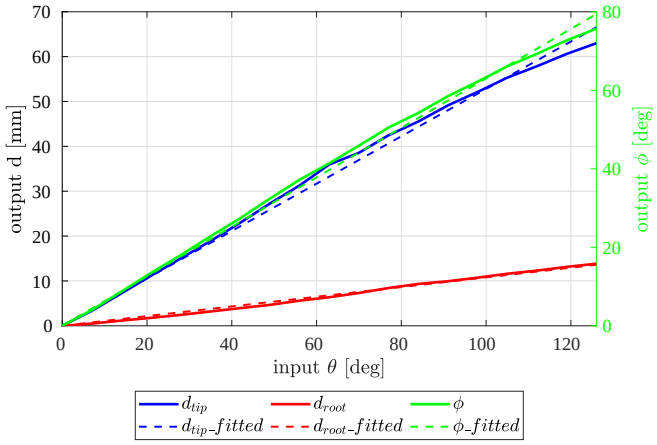


Figure 7: Measured and fitted outputs of the mechanism

d_{root}.

$$\begin{aligned} d_{root} &= |(\boldsymbol{U} - \boldsymbol{U_2}) - (\boldsymbol{L} - \boldsymbol{L_2})|_y \\ &= B_U \sin\psi_7 + U_1 \sin(\psi_3 + \phi_3) \\ &\quad - [B_L \sin\psi_6 + L_1 \sin(\psi_4 + \phi_4)] \end{aligned} \tag{6}$$

The beak opening angle ϕ is the angle between the upper and lower beaks, and can be expressed as:

$$\phi = \phi_3 + \phi_4 \tag{7}$$

As a single-degree-of-freedom mechanism, the key outputs d_{tip}, d_{root}, and ϕ depend solely on the input angle θ. By determining the relationships between joint angles and the input θ, we can precisely control these outputs through servo motor positioning. The mechanism's design allows for adjustment of link parameters to achieve different beak opening ranges and angles, enabling adaptation for various UAV platforms and applications. Table 2 lists the parametric values used in our implementation.

To simplify the output equations, a CAD model of the mechanism was used to measure the outputs of different input angles. The resulting discrete data were plotted, and linear fitting was applied to approximate the corresponding equations. The plotted data and the fitted equations are shown in figure 7. The equations are expressed as below.

$$d_{tip} = 0.5280 * \theta \quad [mm] \tag{8}$$

$$d_{root} = 0.1079 * \theta \quad [mm] \tag{9}$$

$$\phi = 0.5630 * \theta \quad [deg] \tag{10}$$

3 Experiments and Evaluation

The pecking mechanism evaluation consisted of two main testing phases: static force analysis and dynamic flight trials. Static testing measured gripping capabilities across objects of various materials and sizes. Dynamic flight trials evaluated four key manipulation tasks: payload transport, grasping and pulling operations, sample collection, and pecking maneuvers. The experimental methodology focused on mechanism performance assessment under realistic operating conditions.

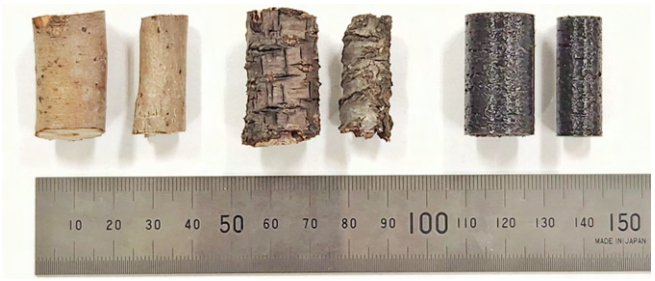


Figure 8: Sample objects for static grasping experiment

3.1 Grasping Force Test

Force measurements using a Newton meter were conducted to quantify gripping capabilities across six test specimens of varying materials and dimensions. The experimental protocol involved rope attachment to specimens, Newton meter coupling, and progressive tension application until grip failure occurred. Video documentation of force readings enabled peak value determination for maximum grip strength analysis. The experiment was performed 3 times for all objects. The objects are shown in Figure 8, and the experiments were conducted as shown in Figure 9. The objects are all cylindrical, and are categorized in 2 sizes and 3 materials. The sizes are 12 mm and 18 mm, and the materials are smooth branch (left), rough branch (center), and soft TPU (right).

The results are shown in Figure 10. In this box-and-whisker plot, the vertical axis represents the maximum gripping force. For each object, the plot shows the range and the mean-maximum gripping force. The mean value is indicated by the black line inside the boxes.

The mean-maximum grasping force was the highest for the 18 mm rough branch at 3.03 N, followed by 2.29 N for the 18 mm soft TPU, 2.06 N for the 12 mm soft TPU, 1.63 N for the 12 mm rough branch, 0.96 N for the 18 mm smooth branch, and lastly 0.93 N for the 12 mm smooth branch.

The results show that larger objects (18 mm diameter)

Length [mm]				Angle [deg]	
S	18.8	U_2	40.0	ψ_1	19.1
H	11.5	L_1	16.0	ψ_2	83.0
Q	15.0	L_2	40.0	ψ_3	111.6
J	25.3	K_B	6.0	ψ_4	24.7
B_L	31.9	K_L	10.0	ψ_5	30.0
B_U	7.6	α	0.7 [-]	ψ_6	92.4
U_1	13.9			ψ_7	147.1

Table 2: Mechanism parametric values

http://www.imavs.org/

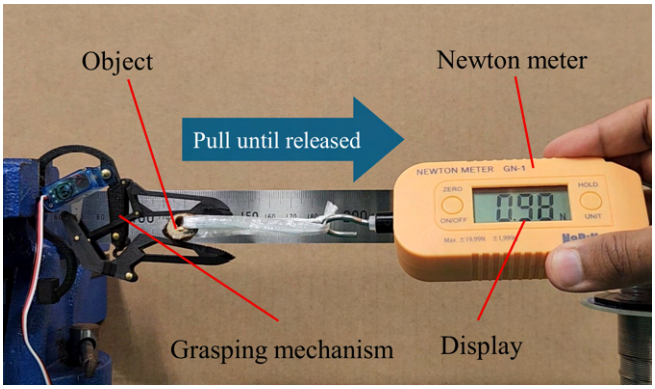


Figure 9: Static grasping experiment

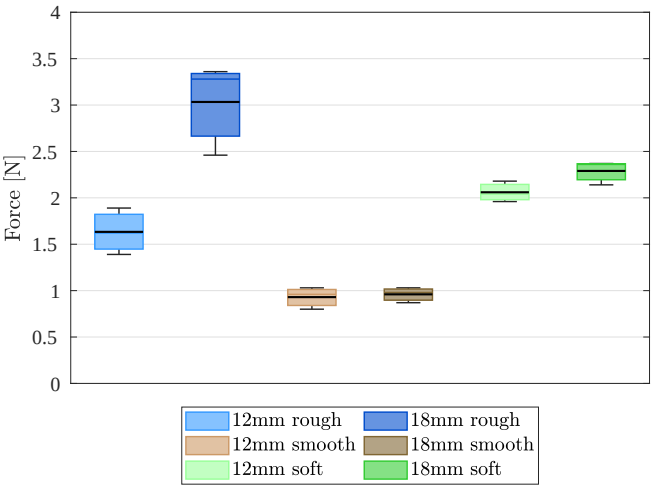


Figure 10: Static grasping experiment result

consistently required more force to dislodge than smaller ones across all materials tested. This difference was most pronounced with the rough branch, moderate with the soft TPU, and minimal with the smooth branch. The increased grip strength on larger objects can be attributed to their greater contact area with the beak surfaces. When gripped, the objects experience slight deformation at the contact points. Larger objects, having more gradual curves, create wider contact areas under the same gripping pressure, leading to stronger overall grip force. This relationship explains the minimal force differential observed with the smooth branch, as its rigid structure resists deformation under the mechanism's grasping force. Notably, the force difference between size variants is greater for the rough branch than for the soft TPU, despite TPU's higher compressibility. This unexpected result stems from surface characteristics: the rough wood's surface protrusions interact mechanically with the grooved beak tips, enhancing grip through physical interlocking rather than material deformation alone.

Analysis of force measurement variability reveals that smooth-surfaced objects (smooth branch and soft TPU) exhibited 60% less variation in maximum grasping force compared to the rough branch. While rough surfaces generated higher peak grip forces, smooth surfaces demonstrated superior consistency in force application, suggesting enhanced operational reliability and repeatability.



Figure 11: Piece of a plant as a payload (6.7 g)

The minimum measured grasping force of 0.93 N exceeded the mechanism's weight by a factor of 5.96, demonstrating robust gripping capability across the full range of test specimens, independent of material properties, surface characteristics, and geometric variations.

3.2 Flight Experiments

Following validation of the mechanism's static grasping capabilities across diverse object types, dynamic flight testing was conducted to evaluate performance in realistic operational scenarios. Four distinct manipulation tasks were executed to assess aerial manipulation capabilities. The experimental platform incorporated a nonlinear disturbance observer-based robust controller [10]. Mounted on the Nimble+ platform (figure 1), the mechanism responded to external control inputs throughout the aerial manipulation sequences.

3.2.1 Payload Transportation Task

In this experiment, the capability to successfully carry a payload to a different location while flying was tested through a payload transportation task. First, the payload shown in Figure 11 is placed to be grasped by the mechanism. After that, the drone starts by hovering, and then transitions to start maneuvering around, demonstrating carrying a payload to a different location. The flight experiment is shown in Figure 12.

Here, the mechanism's ability to transport payloads during flight was tested using the sample shown in Figure 11. The test sequence began with positioning the payload for grasping. Once secured, the MAV took off into a stable hover before executing a series of maneuvers to demonstrate payload transport capability. Figure 12 shows the key stages of this flight experiment.

The experiment demonstrated stable payload transport throughout the flight sequence, with the mechanism maintaining a secure grip while the MAV performed various maneuvers. These results confirm the design's practical utility for aerial payload transport applications.

http://www.imavs.org/

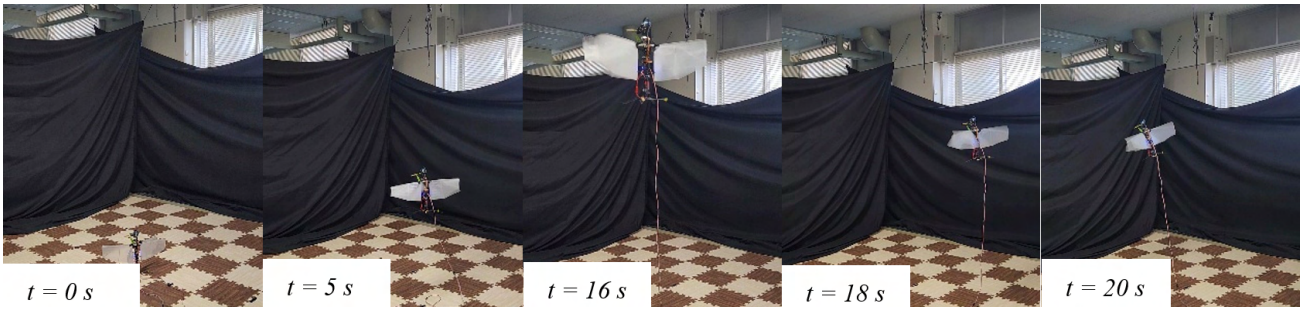


Figure 12: Payload Transportation task

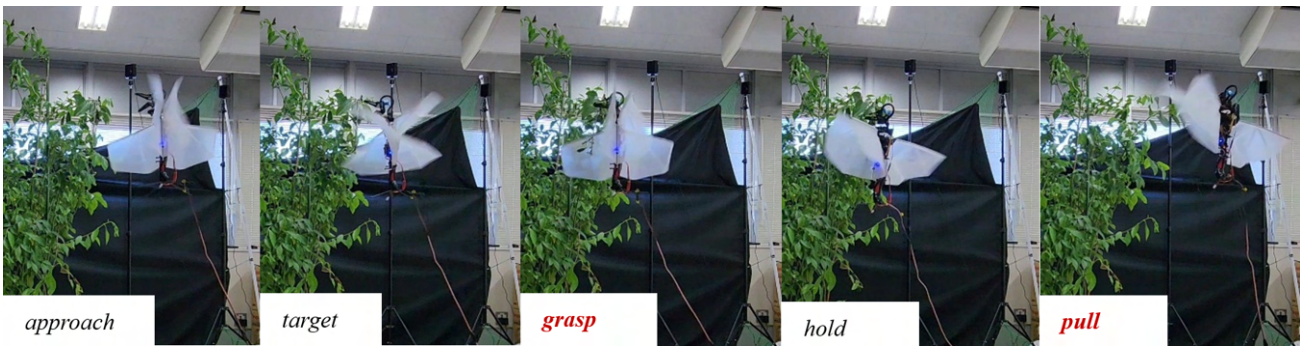


Figure 13: Grasping and Pulling task

http://www.imavs.org/

3.2.2 Grasping and Pulling Task

To simulate real-world tasks, system's ability to grasp and manipulate objects before transportation was tested. For this evaluation, a potted plant was positioned at a set distance from the takeoff point. After takeoff, the MAV approached the plant in a controlled manner. When positioned near the target leaf, the beak mechanism was opened. The MAV then moved closer for precise positioning, allowing the beaks to close and secure the leaf. Once grasped, the MAV tested the grip strength by pulling in different directions. The flight experiment is shown in Figure 13.

The approach maneuver execution resulted in successful leaf acquisition. Multi-directional loading tests demonstrated stable grip integrity, with maintained hold observed under maximum thrust conditions. These results validate system capabilities for manipulation tasks requiring sustained force application.

3.2.3 Grasping and Sample Collection Task

The complete sample collection sequence consisted of approach, grasp, detach, and transport operations. The experimental methodology followed procedures outlined in Section 3.2.2, targeting branch sample retrieval and return to the initial position. Double-sided tape secured the test specimen to an existing branch. Figure 14 documents the complete flight sequence.

The experiment demonstrated successful sample retrieval and landing operations. The demonstrated collection capabilities, while successful in controlled conditions, indicate the necessity for an integrated cutting mechanism in practical field applications for direct vegetation sampling.

3.2.4 Pecking Task

The final experimental phase evaluated forward force application capabilities through pecking motions. The test setup employed a suspended banana as the target object. Experimental procedures consisted of target approach followed by pecking motion execution. Results demonstrated flexible neck effectiveness in impact force absorption and primary structure protection.

The impact force proved sufficient for target displacement, while the neck's compliance characteristics enabled stable flight maintenance throughout the impact sequence.

4 Conclusion

A bio-inspired aerial manipulation system based on avian Cranial Kinesis principles is introduced in this study. The proposed single-degree-of-freedom design replicates key skeletal movements of bird skulls, achieving substantial beak articulation and grip force while maintaining minimal mass. Through static force analysis and dynamic flight trials, successful execution of diverse manipulation tasks was demonstrated, including object transport, grip-and-pull operations,

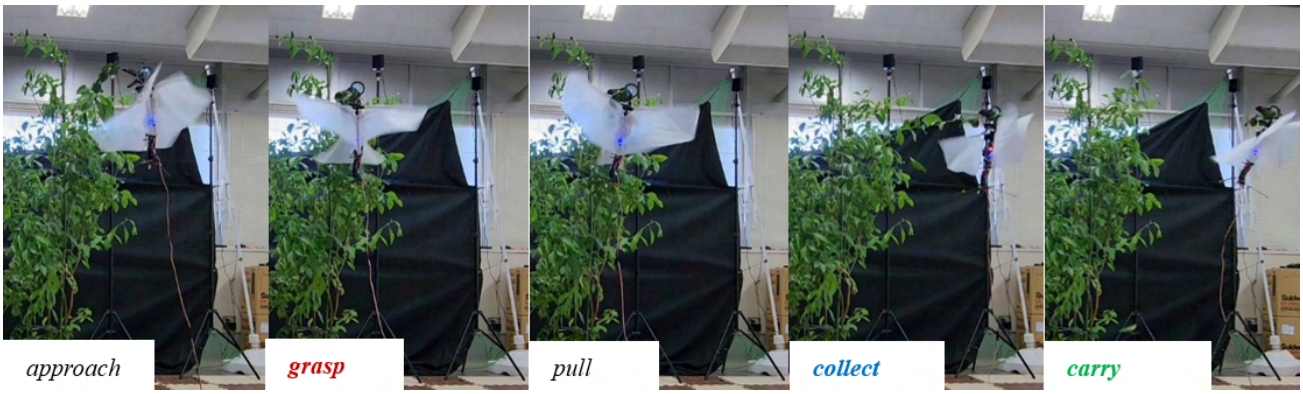


Figure 14: Grasping and Sample Collection task

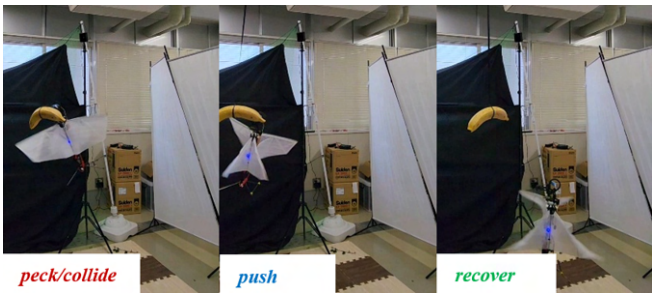


Figure 15: Pecking task

and pecking maneuvers. Future development directions include design optimization, integration of cutting functionality within the current kinematic framework, and implementation of advanced control architectures for enhanced operational capabilities.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number JP23K13274 (Grant-in-Aid for Early-Career Scientists)

References

[1] Raphael Zufferey, Jesus Tormo-Barbero, D. Feliu-Talegón, et al. How ornithopters can perch autonomously on a branch. *Nature Communications*, 13(7713), 2022.

[2] Ahmad Hammad, Mehmet Süer, and Sophie F. Armanini. A lightweight bioinspired sma-based grasping mechanism for flapping wing mavs. *Biomimetics*, 10(6), 2025.

[3] Krispin Broers and Sophie Armanini. Repeatable energy-efficient perching for flapping-wing robots using soft grippers, 09 2024.

[4] Saeed Rafee Nekoo, Daniel Feliu-Talegon, Raul Tapia, Alvaro C. Satue, Jose Ramiro Martínez-de Dios, and Anibal Ollero. A 94.1 g scissors-type dual-arm cooperative manipulator for plant sampling by an ornithopter using a vision detection system. *Robotica*, 41(10):3022–3039, 2023.

[5] Rodrigo Gordillo Durán, Raul Tapia, Saeed Rafee Nekoo, José Ramiro Martínez-de Dios, and Anibal Ollero. Flapping-wing flying robot with integrated dual-arm scissors-type flora sampling system. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[6] Sahar Sadeghi Kordkheili, Antonio Gonzalez-Morgado, Saeed Rafee Nekoo, Begoña C. Arrue, and Anibal Ollero. Optimal control of dual arm manipulation for flapping-wing robots in the post-perching phase. In *2025 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 155–161, 2025.

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

[8] William Hoese and Mark Westneat. Biomechanics of cranial kinesis in birds: Testing linkage models in the white-throated sparrow (zonotrichia albicollis). *Journal of Morphology*, 227:305–320, 03 1996.

[9] Preeda Meekangvan, Alan A. Barhorst, Thomas D. Burton, Sankar Chatterjee, and Lawrence Schovanec. Nonlinear dynamical model and response of avian cranial kinesis. *Journal of Theoretical Biology*, 240(1):32–47, 2006.

[10] A. Asignacion S. Hussain, T. Matsuda and S. Suzuki. Nonlinear disturbance observer-based robust control for flapping-wing aerial manipulators. In *submitted to IEEE Trans of Mechatronics*, 2025. Under review.

http://www.imavs.org/

Flight Pattern and Experimental Flights of a Quadcopter Drone at the Edge of a Smoke Plume

Sacha De Sousa[1,2]*†, Maya Pivert[1]‡†, Renaud Kiefer[1]§, Sylain Durand[1] and Édouard Laroche[2]
[1]Univ. Strasbourg, CNRS, INSA Strasbourg, ICube, UMR 7357, 67000 Strasbourg, France
[2]Lycée des métiers Louis Couffignal, 67000 Strasbourg, France

http://www.imavs.org/

ABSTRACT

Measuring air quality is a burning challenge, and an increasing number of drones are now equipped with pollution sensors to detect airborne pollutants. This paper presents a trajectory tracking strategy that allows the drone to fly along the edge of the plume, preventing it from entering the plume where heat and gas concentrations are high. Algorithms developed in Python were tested in simulation with virtual plumes based on emulated GPS data, then validated experimentally in the field and during a real fire using an onboard particle sensor embedded on a quadcopter.

1 INTRODUCTION

One of the most significant industrial fires in France in recent years remains that of *the Lubrizol and Normandie Logistique* sites, which occurred on September 26, 2019, near Rouen. This large-scale disaster ignited a hydrocarbon pool covering more than 20,000 m^2, releasing a massive plume of black smoke over the city. More recently, on April 9, 2025, a battery recycling plant exploded in Kilwinning, Scotland. In the face of such events, understanding the composition of these plumes becomes a crucial issue: it is not only a matter of improving risk management, but also of informing and protecting populations.

When a fire breaks out, large quantities of fine particles (PM) and polluting gases – such as nitrogen dioxide (NO_2), sulphur dioxide (SO_2) and carbon dioxide (CO_2) – are released into the atmosphere [1]. Depending on the fire's origin, other gases like hydrogen fluoride (HF), hydrogen chloride (HCl), and similar compounds may also be emitted, particularly in the case of industrial fires. Exposure to these substances can have deleterious effects on the health of local populations, depending on the duration and concentration of exposure. Faced with these risks, the emergency services must rapidly assess the situation to establish a safety perimeter and, if necessary, organize the evacuation of areas at risk.

*Email address: sacha.desousa@hotmail.fr
†Both authors contributed equally to the conception and writing of the article.
‡Email address: maya.pivert@insa-strasbourg.fr
§Email address: renaud.kiefer@insa-strasbourg.fr

In such contexts, drones have been extensively deployed to monitor smoke plumes and the dispersion of pollutants. The analysis of fire-related smoke, however, raises two major challenges: should the measurements be conducted indoors, as in the case of an apartment fire, or outdoors? In the former case, the drone must operate directly within the smoke and therefore requires an architecture capable of withstanding extreme temperatures. The FireDrone project [2] addressed this issue by introducing a thermally resistant UAV, equipped with insulating materials and a biologically inspired phase-change cooling system, enabling safe operation under harsh conditions.

For outdoor smoke analysis, such as in wildfires or large-scale industrial accidents, plumes can be transported over several kilometers. To tackle these challenges, numerous research initiatives—supported in France in particular by the French National Research Agency (ANR) [3, 4]—have focused on developing innovative approaches for measuring and mapping air pollution using fleets of drones. [5] proposed a ConvLSTM-based model coupled with data assimilation to map smoke dynamics in real time, with adaptive recalculation of drone fleet trajectories. Similarly, [6] developed a cooperative reinforcement learning approach to optimize multi-drone coordination and produce reliable plume maps.

In this article, we focus on large-scale industrial fires, for which the development of effective plume-mapping strategies is essential. In the literature, two main classes of strategies for mapping pollutant clouds can be identified: those in which the drone penetrates the cloud, and those in which it navigates along its boundary.

The first family consists of flying the drone inside the cloud to collect measurements at the heart of the plume. For example, [7] proposes strategies such as the rosette trajectory, which involves 2D exploration by several internal passes, and the oblique cone spiral, characterized by a helical movement within the cloud. Other works, such as [8], use swarms of drones to detect, explore, and map the boundaries of the cloud using coordinated dispersion.

However, in the specific case of fire plumes, the high heat and high concentrations of pollutants make it risky to fly directly through the cloud, both for the drone and its sensors, that risk being saturated or damaged. Rather than using reinforced UAVs and sensors [2], a parsimonious approach is favored by limiting flight to the periphery of the plume.

We are interested in the second family of cartography

http://www.imavs.org/

strategies, in which the drone evolves only on the border of the cloud. In [7] notably, two approaches are distinguished: the "Trinity" strategy, which consists of loop trajectories around the periphery of the cloud ; and the "Lace" strategy, where the drone follows arcs along the border of the cloud (see Figure 2) similar to the methods presented in [8, 9]. The adopted strategy is derived from the latter.

More specifically, this study presents the development of an autonomous system built around a quadcopter drone equipped with a particle sensor, designed for real-time mapping of smoke plumes and for monitoring pollutants along the plume boundary. Following the description of the drone prototype and its onboard detection system (Section 2), the article examines the proposed flight control strategy (Section 3), details the experimental campaigns (Section 4), and discusses the results obtained (Section 5).

2 Architecture

2.1 Drone Prototype

The prototype used in this study is a quadcopter based on the X500 kit from Holybro. The carbon fiber frame adopts an X configuration. It measures 50 cm in diameter and has an empty weight of 978 g. The kit includes all the components necessary for assembly: a Pixhawk 4 autopilot, four brushless motors, four electronic speed controllers (ESCs), a PM07 power distribution board, a GPS module, propellers, a radio telemetry module, a FrSky XR8 receiver, as well as power and communication cables. Manual control of the drone is provided by a FrSky Taranis X7ACCESS remote controller, which transmits instructions to the Pixhawk 4. To enable autonomous missions, a Raspberry Pi 3B+ mini-computer is also onboard. It communicates with the Pixhawk via a USB connection and hosts the adaptive flight algorithms. Additionally, the Raspberry Pi receives sensor data through an SPI connection with an Arduino Mega, which serves as an interface for collecting environmental measurements.

2.2 Onboard Measurement System

The measurement module mounted on the drone integrates the following components:

- An OPC-R2 20 particle sensor,
- A GSM communication module,
- A GPS module,
- A SD card reader,
- An Arduino Mega.

The OPC-R2 sensor, which operates based on light diffraction principles, enables the detection and classification of airborne particles. The data collected by the sensor are processed by the Arduino Mega, and then transmitted to an online database via a 2G connection provided by the GSM module. In parallel, the measurements are geolocated using the GPS module and stored locally on an SD card.

The collected data are also sent to the onboard Raspberry Pi for further processing via an SPI connection. The entire system is powered by a 4S 3700 mAh Tattu LiPo battery. An overview diagram of the system is presented in Figure 1.

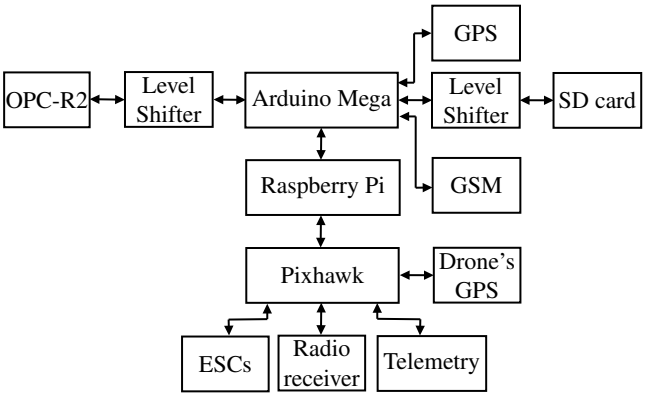


Figure 1: Architecture of the measurement system.

3 Plume-Edge Tracking Strategy

3.1 Navigation Strategy: the 'Lace' Approach

As part of our application, the adopted navigation strategy is inspired by the 'Lace' method described in the paper [7] and illustrated in Figure 2. This strategy is based on a trajectory pattern composed of successive circular or elliptical arcs, alternating inside and outside the area to be mapped. When a particle concentration threshold is exceeded, the drone initiates a trajectory along the boundary of the cloud, alternating between approach and bypass phases.

This strategy can be innovatively reimagined by redefining its core approach. For example, one novel adaptation involves exclusively performing exterior trajectories, see Figure 3, thereby circumventing sensor saturation that would otherwise result from the high particle concentrations found at the core of the cloud. Similarly, the shape of the trajectories can be adapted: instead of following a perfect circle, an ellipse or a polynomial curve can be used to better match the actual shape of the cloud.

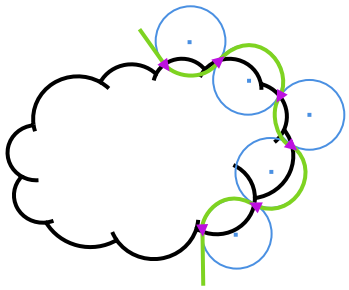


Figure 2: Lace adaptive flight pattern derived from [7].

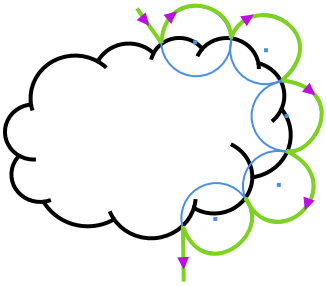


Figure 3: The 'Petals' strategy used in this work.

3.2 *Strategy Process*

The Lace strategy is divided into several steps:

1. Initial Approach: the drone flies in a straight line toward the estimated position of the cloud.
2. Boundary Detection: as the drone enters the cloud (defined by a particle concentration threshold), it stops its forward movement.
3. Looped Trajectories: the drone then performs circular or elliptical trajectories around the concentration zone each time it detects the cloud boundary.
4. Mission Completion: Once the mission-ending conditions are met (e.g., low battery), the drone autonomously concludes the tracking of the cloud.

Simulations were conducted using the QGroundControl software, which allows for aerial visualization of the drone's trajectory.

3.3 *Algorithmic Architecture*

The proposed control algorithm is represented by the flowchart in Figure 4. The adapted 'Lace' strategy – afterwards called 'Petals' – workflow is as follows:

- Takeoff preparation: switch to "STABILIZE" mode while awaiting user instruction.
- Takeoff: Switch to "GUIDED" mode, arm the drone, and ascend to the target altitude.
- Navigation toward the cloud: fly toward the predefined GPS coordinates corresponding to the suspected area.
- Position hold and trajectory calculation: In "LOITER" mode, the drone holds its position and computes the GPS waypoints forming the elliptical trajectory.
- Trajectory execution: move in straight lines between the GPS waypoints of the ellipse.
- Boundary loss: if the cloud boundary is lost, adjust the ellipse parameters to relocate it. Loss of boundary is detected when the drone has traveled through all the GPS points of the current ellipse. At that point, its axes are multiplied by 1,3 for the next ellipse, and this process repeats as long as the boundary remains undetected. As soon as the boundary is found, the ellipse returns to its original size. Additionally, with each iteration, the ellipse's angle is shifted by 0,7 radians to improve the search for the cloud's boundary.
- End of mission: once the mission-ending conditions are met, the drone executes an exit procedure.

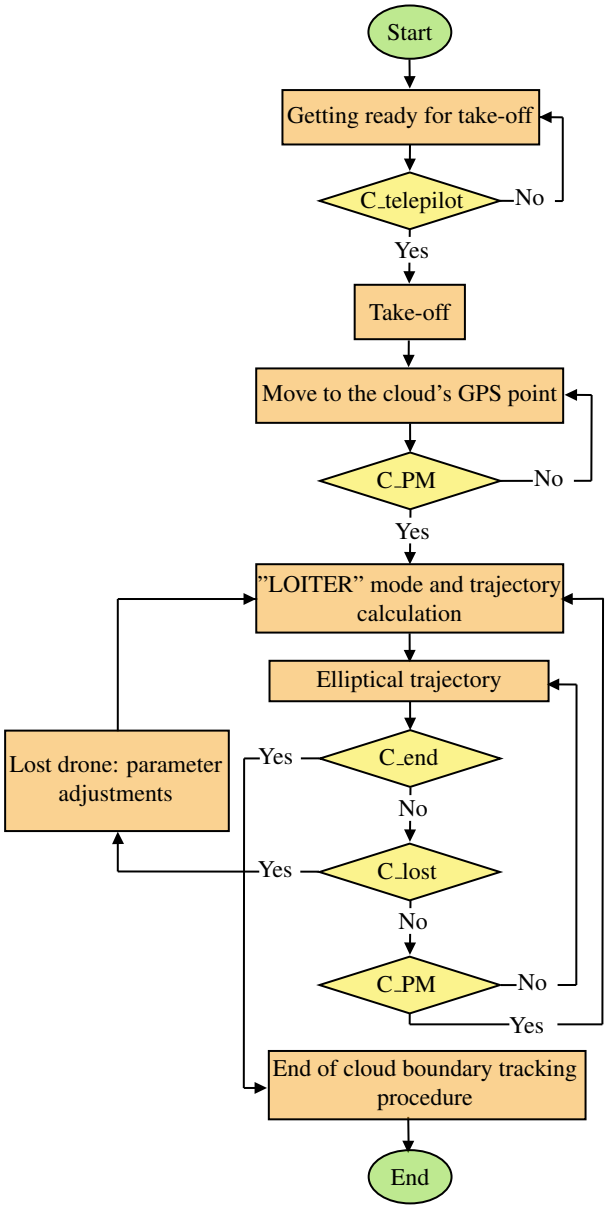


Figure 4: Flowchart of the adaptive flight pattern algorithm (Processes are shown as orange rectangles, and transition conditions as yellow diamonds).

The transition conditions between each of these steps are as follows:

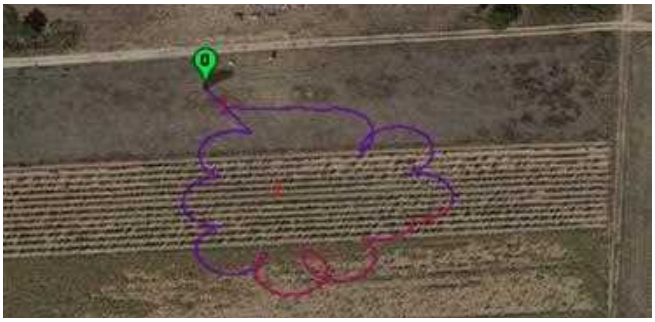
- C_pilot: the drone takes off once the user activates "AUTO" mode.
- C_PM: when the particle concentration (PM1 or PM2.5) exceeds 70 μg/m^3, the drone considers that it has entered the cloud.
- C_end: if the battery level becomes critical, the mission is interrupted to ensure the drone's safety.
- C_lost: this condition is triggered if the drone completes an elliptical trajectory without detecting any variation in the C_PM condition, indicating a loss of the cloud boundary.

http://www.imavs.org/

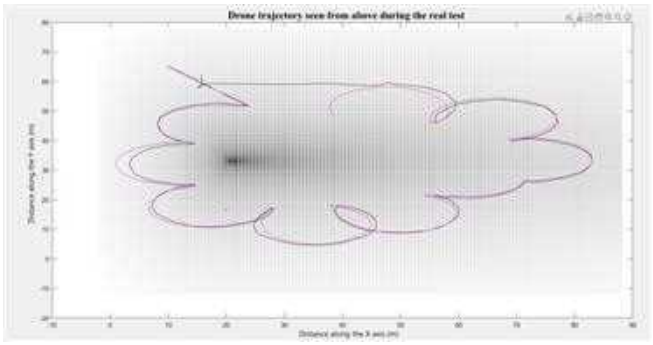
4 Experiments and Results with the Particle Sensor

4.1 Simulations and Experiments on an Emulated Sensor

In order to validate the flight strategy, initial tests were conducted using a simulated sensor. This device emulates the concentration measurements of the artificial cloud based on the drone's GPS position. The main objective of this first flight test campaign was to evaluate the "Petals" flight strategy without using the actual particle sensor, by following the planned trajectory along the boundary of a cloud artificially



(a) GPS position: comparison of the theoretical trajectory of the drone obtained through simulation (in purple) with the trajectory measured during the flight test (in black).



(b) 2D trajectory of the drone around the cloud generated during the mission.

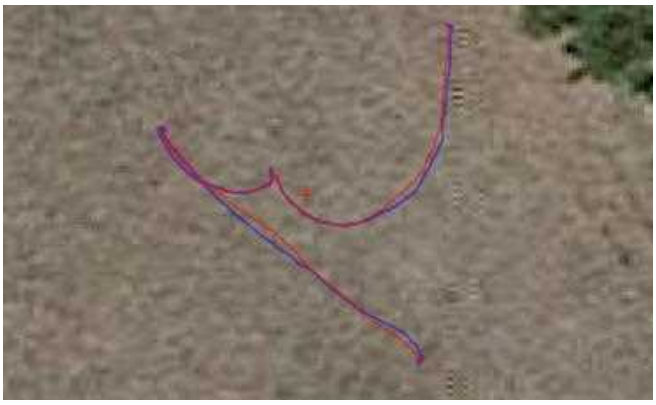
Figure 5: Flight carried out during the mission at the model aircraft field in Brumath, Alsace, France.

generated by the Raspberry Pi. This approach aimed to identify any weaknesses in the strategy and suggest areas for improvement before deploying it with an operational particle sensor. During the mission, the drone followed the trajectory shown in Figure 5 at the model aircraft field, under favorable weather conditions (sunny weather and no wind). A good fit between the two curves can be observed when comparing the simulated and experimental trajectories (Figure 5a). This closeness indicates that the simulation results accurately reflect the drone's actual behavior during real-world testing.

This test made it possible to validate, through field experiments, the proper functioning of the adaptive flight scheme by demonstrating its ability to accurately reproduce the expected trajectory as simulated. The main issues identified and resolved during this phase concerned sudden altitude variations as well as an irregular drone trajectory, particularly during elliptical paths. In conclusion, these experimental tests confirmed the effectiveness and robustness of the 'Petals' adaptive flight algorithm, developed in Python. The autonomous strategy for tracking pollutants at the edge of a smoke plume was thus validated in a real environment using a simulated sensor. The next step involved testing this approach on a real smoke plume, using a drone prototype equipped with an onboard particle sensor.



(a) Smoke cloud produced by the smoke bomb.



(b) Trajectory during a smoke bomb test.

Figure 6: Experimental test on smoke bombs in a field.

4.2 Experimental Results on a Smoke Bomb

Before performing a test on an actual fire, an initial way to ensure that the adaptive flight strategy works with the particle sensor is to first test it on smoke puffs generated by a smoke bomb. The smoke bomb generated a small cloud (Figure 6a) lasting only about 2 minutes. This is too short to fully map and follow the boundary, but sufficient to verify the correct triggering of the program's different processes.

http://www.imavs.org/

http://www.imavs.org/

The threshold set for the smoke plume boundary is 70 μg/m^3 for PM1 and PM2.5. PM1 and PM2 are particulate matter sizes, with PM1 being particles smaller than 1 micrometer and PM2.5 being particles smaller than 2.5 micrometers. This choice was made from experiments conducted using smoke from a barbecue and a smoke bomb. The trajectory followed by the drone during the test with a smoke bomb is shown in Figure 6b. During this experimental test, the weather was sunny with a northwest wind at 10 km/h and occasional gusts. This test with a smoke bomb confirmed that the program was functioning correctly, enabling subsequent testing on the smoke plume from a controlled fire chamber. Indeed, the drone stops and switches to "LOITER" mode when the threshold is exceeded, then performs elliptical trajectories along the edge of the smoke puffs.

4.3 Experimental Results on a Fire Training Container

The final experimental test was in sunny weather with a East/Northeast wind. The objective of this test was to evaluate the adaptive flight pattern over a smoke plume originating from a fire training container. For this purpose, a member of The Bas-Rhin Fire and Rescue Service (SIS67) ignited a fire using wooden pallets inside a container, as shown in Figure 7a. A GPS point at the center of the plume was defined, and the input parameters for the algorithm were as follows:

- An ellipse radius of 5 meters: a major axis of 6 m (1.2 × 5 m) and a minor axis of 4 m (0.8 × 5 m)
- A flight altitude of 10 meters
- 35 GPS waypoints generated for each ellipse
- An angle between successive ellipses of 85 °C
- A maximum flight speed of 1.4 m/s during the elliptical path
- A GPS point distance factor of 2 during the elliptical trajectory
- Ellipse orientation set at 45 °C, given the wind from the East/Northeast
- PM1 and PM2.5 entries threshold into the plume: 70 $\mu g/m^3$
- PM1 and PM2.5 exits threshold from the plume: 70 μg/m^3

In Figure 7b, the drone's flight path over the fire training container area is shown. Figure 7c shows the drone's trajectory re-plotted in MATLAB, both in a 3D perspective. The drone initiates its strategy approximately 25 meters from the fire source.

5 ANALYSIS OF THE RESULTS

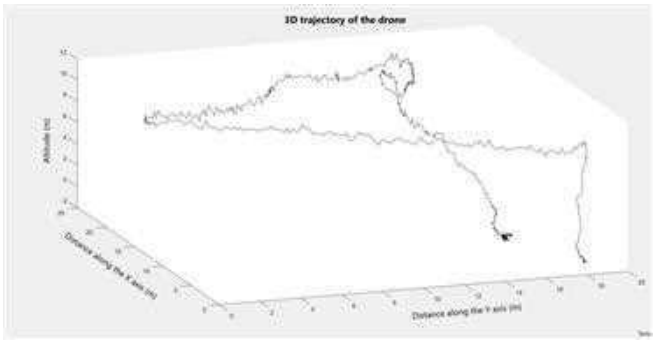
Several test campaigns were conducted using the OPC-R2 particle sensor to monitor real smoke plumes, with continuous readings of PM concentrations. Preliminary trials with smoke bombs were instrumental in calibrating the detection threshold and resolving issues related to particle concentration readings. As detailed in Section 4.2, these tests led to a functional integration of the sensor with the adaptive flight algorithm: the drone successfully followed an adaptive flight pattern around the smoke generated by the bombs. However, smoke bombs present certain limitations – their plumes are low, not very dense, and short-lived (less than two minutes). These constraints make it difficult to fully evaluate the flight strategy in such conditions.



(a) Smoke plume from the fire training container.



(b) GPS drone trajectory during the fire training container test (the drone trajectory is marked in yellow, the green arrow indicates wind direction, and the red cross marks the fire source).



(c) 3D trajectory flown by the drone around the smoke plume.

Figure 7: Experimental test on a fire training container in the industrial zone of Drulingen, Alsace, France.

To overcome these limitations, a final test on a real fire-generated smoke plume was carried out, as described in Section 4.3. The results were promising: the drone correctly identified and tracked the edge of the plume using the adaptive trajectory. Nonetheless, the flight was brief, as the drone had to be retrieved early due to collision risks.

One challenge encountered was the drone's ability to locate the plume at the start of the mission. The predefined GPS point, intended to lie within the plume, was sometimes

outside the actual smoke zone due to inaccurate positioning, overly strict PM detection thresholds, or suboptimal flight altitude. These three parameters – initial GPS point, PM threshold, and flight altitude – require fine-tuning to ensure proper activation of the adaptive strategy. Future trials are needed in safer environments with fewer obstacles, along with better control over parameter settings. Overall, these tests confirm the functionality of the adaptive flight algorithm for mapping and tracking pollutants along smoke plume boundaries.

6 Conclusion

This work develops and evaluates an autonomous drone system for tracking airborne pollutants along smoke plume edges. A trajectory-following strategy, based on adaptive path planning, was developed, simulated, and experimentally validated through a series of tests using both smoke bombs and a controlled fire training container. A total of 27 test flights were conducted over 9 days, amounting to 1 hour and 5 minutes of total flight time with the prototype. The result is a first operational prototype of an autonomous quadcopter equipped with a functional OPC-R2 particle sensor. The drone is capable of detecting variations in PM1 and PM2.5 concentrations and autonomously adapting its trajectory to follow the boundaries of a smoke plume.

However, this prototype remains an initial proof of concept, and multiple avenues for improvement are identified for future development. Algorithmically, enhancements could include replacing elliptical trajectories with polynomial path generation for better adherence to irregular plume shapes, implementing more robust boundary-loss recovery strategies, and enabling 3D mapping of the smoke plume. The implementation of performance criteria for flight paths is also part of the ongoing work.

To date, testing has been limited to small-scale conditions, covering distances of approximately 100 meters, whereas the ultimate objective is to map large-scale smoke plumes extending over several kilometers. To achieve this goal, flight endurance will need to be improved through higher-capacity UAV platforms and more robust, integrated electronic architecture. Our team is currently developing a VTOL-type drone designed to enable longer flight durations, thereby improving flight performance while reducing battery consumption. Future sensor integration could also involve the addition of gas sensors, humidity sensors, and temperature probes to enrich the environmental data collected. The long-term goal is to develop a deployable system that could assist firefighting units in real-time plume tracking and pollutant monitoring during active fire events.

Acknowledgements

This project has received financial support from the CNRS through the MITI interdisciplinary programs through its exploratory research program. We would like to express our gratitude to Lieutenant-Colonel Patrice PETIT (SIS67) for his guidance and supervision, particularly for his expertise and advice regarding firefighting operations and fire-related aspects of the project.

We also thank Dr. Stéphane LE CALVE (CNRS, ICPEES laboratory) for his scientific input in atmospheric chemistry and for his expert recommendations regarding the selection and use of particulate matter sensors.

References

[1] Benjamin Truchot, Jauris Joubert, and Karen Perronnet. Recensement des substances toxiques susceptibles d'être émises par un incendie. Technical report, Institut National de l'Environnement Industriel et des Risques, Verneuil en Halatte, June 2023.

[2] David Häusermann, Sam Bodry, Fabian Wiesemüller, Aslan Miriyev, Severin Siegrist, Fan Fu, Sabyasachi Gaan, Matthias M. Koebel, Wim J. Malfait, Shanyu Zhao, and Mirko Kovač. FireDrone: Multi-Environment Thermally Agnostic Aerial Robot. *Advanced Intelligent Systems*, 2023.

[3] Gregory Roberts. Réseaux pour l'étude de l'entrainement et la microphysique des nuages par l'exploration adaptative, 2017.

[4] Gautier Hattenberger. Déploiement d'une flotte de drones pour le suivi temps-réel de pollution atmosphérique accidentelle, 2021.

[5] Ichrak Mokhtari, Walid Bechkit, and Herve Rivano. A generic framework for monitoring pollution plumes in emergencies using UAVs. July 2021.

[6] Mohamed Sami Assenine, Walid Bechkit, Ichrak Mokhtari, Herve Rivano, and Karima Benatchba. Cooperative Deep Reinforcement Learning for Dynamic Pollution Plume Monitoring Using a Drone Fleet. *IEEE Internet of Things Journal*, January 2023.

[7] Titouan Verdu, Nicolas Maury, Pierre Narvor, Florian Seguin, Gregory Roberts, Fleur Couvreux, Grégoire Cayez, Murat Bronz, Gautier Hattenberger, and Simon Lacroix. Experimental flights of adaptive patterns for cloud exploration with UAVs. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2020.

[8] M.A. Kovacina, D. Palmer, Guang Yang, and R. Vaidyanathan. Multi-agent control algorithms for chemical cloud detection and mapping using unmanned air vehicles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, September 2002.

[9] G S Mani. Mapping contaminated clouds using UAV — A simulation study. In *2013 Annual IEEE India Conference (INDICON)*, December 2013.

http://www.imavs.org/

Dielectric Barrier Discharge Plasma Actuators to Mitigate Flow Separation

Maggie Gaskell, Robin Krasinkiewicz, Janet Liu, Noah Miggiani, Charlie Smith, Annie Wu, Haowen Yan, Aidan Woods
Queen's University Aerospace Design Team, Kingston, Ontario, Canada

http://www.imavs.org/

ABSTRACT

The following study investigates dielectric barrier discharge (DBD) plasma actuator enhancement for fixed wing micro air vehicles (MAVs) to mitigate airflow separation and increase the stall angle. Airflow separation optimization allows for controlled flight at low Reynolds numbers for low speeds and high altitudes. The following work implements simulations in computational flow dynamics (CFD) software, including Simscale and Openfoam, to determine the location along the chord length of a NACA0012 airfoil where flow separation occurs for various angles of attack. The resulting data determines that the optimal location for the DBD plasma actuators is with their frontmost edges coincident with the airfoil's leading edge. Transient simulations in OpenFOAM with and without a plasma actuator validate the effectiveness of the actuator location with comparison to non-actuated values from previous studies. The actuator models are developed using momentum sources and Python code was used to graph the data and determine average coefficient and momentum values. Comparisons between the resulting coefficients and those in previous studies determine that at a Reynolds number of 53,000 the actuators consistently increase the lift and decrease the drag coefficients when placed at the leading edge of the airfoil.

1 INTRODUCTION

Flow separation occurs when the boundary layer of air on a wing surface detaches from the wing, causing an increase in parasitic drag and a decrease in lift. Separation often occurs when the wing is subject to adverse pressure gradients or high angles of attack, and in extreme cases can result in a transition from laminar to turbulent flow, leading to aircraft stall. After flow separation occurs, the boundary layer can reattach to the wing further down the chord length, resulting in a separation bubble, in which backflow can occur [1] . Laminar flow, which occurs at low Reynolds numbers (R_e), is more prone to flow separation than turbulent flow. As demonstrated by Equation 1, low R_e flow can occur due to low air density (ρ), low airspeed (v), low chord length (L), or high viscosity (μ). At extremely low R_e, the boundary layer may not reattach after separating, further increasing flight difficulties [2].

$$R_e = \frac{\rho v L}{\mu} \tag{1}$$

Due to the disadvantages of low R_e flow, low speed and high density-altitude flight is often impractical. These limitations are especially prominent in MAVs due to short chord lengths [3] . This narrows the range of altitudes and speeds at which most aircraft can achieve flight, limiting their effectiveness in applications such as flight at the high density-altitudes of Earth's stratosphere, on Mars [3], or low speed flight required in urban environments, such as in urban military combat or search and rescue scenarios [4]. Another limited application is the flight of aircraft used to combat wildfires, due to high density-altitudes caused by the extreme heat.

Flow separation can be mitigated using flat-plate airfoils, as their performance generally does not vary with R_e [2] , or riblets, due to their ability to suppress laminar separation bubbles [5]. However, such methods have notable downsides. Flat-plate airfoils are widely outperformed by thicker airfoils in most applications and thus lack versatility, as they are only useful in low R_e settings [2]. Riblets primarily work to reduce skin-friction drag and thus are not effective on objects with drag forces dominated by pressure drag [6].

An emerging, effective way to mitigate flow separation is the implementation of DBD plasma actuators. They present similar benefits to flat-plate airfoils or riblets, without compromising other aspects of the aircraft's performance. However, actuators introduce other complexities including high-power consumption [7]. As shown in Figure 1, plasma actuators generate a plasma discharge on a wing surface, inducing a body force on the surrounding air. This causes the boundary layer air to accelerate, helping to maintain its attachment to the wing surface, thus delaying or preventing flow separation. As a result, plasma actuators can suppress flow separation, enhance the lift-drag ratio, and increase the maximum stall angle of a wing [8]. DBD plasma actuators offer several advantages over other types, such as plasma jets, including the absence of moving parts, simplicity, and control at high frequencies [9]. Successful implementation of DBD plasma actuators could mitigate the limitations of flight in laminar flow conditions and broaden the uses of fixed-wing aircraft.

For the purposes of this investigation, a NACA0012 airfoil was selected as a base for simulating the non-actuated and actuated flow around the wing due to its abundant use

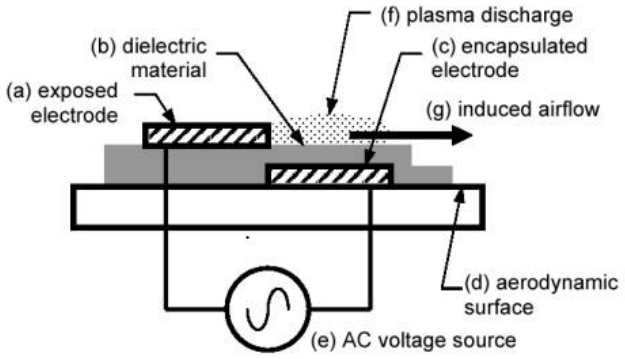


Figure 1: Schematic of a DBD plasma actuator [10].

in previous studies. Known for its well-documented stall behaviour and simple geometry, the NACA0012 airfoil is commonly used in aircraft and aerodynamic surface applications [11].

2 MATERIALS AND METHODS

To determine if a plasma actuator can cause delay in flow separation and increase the maximum lift coefficient, baseline stall angles without actuator influence are required for comparison. Stall angles are specific to airfoil geometry and R_e, therefore pre-existing studies on NACA0012 airfoils with R_e near the general range of MAVs (near 10,000 to 100,000) were investigated. Through experimentation and simulations, a study determined the stall angles of the NACA0012 near the desired R_e of 53,000 [12]. Thus, this R_e was selected for baseline simulations to record values for comparison to validate simulation parameters.

The range of simulated angles of attack, spanning from $4°$ to $16°$, correspond with those used in similar studies to determine if the implementation of actuators increased stall angle. The continual increase of the time-averaged lift coefficients past $10°$ - $11°$ would indicate that the plasma actuator effectively delayed flow separation, resulting in increased maximum lift [12]. Simulations are performed at a R_e of 53,000, and at increments of $0.5°$ determined to capture delayed flow separation without running redundant simulations.

The simulations aim to determine the optimal location of DBD plasma actuators on the wing by determining the locations of flow separation, thereby optimizing their effectiveness in preventing it. Simulations following this determination ran with the actuator placed at the identified location to evaluate its effectiveness.

Simscale was used with the goal of determining the angles of attack at which flow separation occured on the NACA0012, and the location along the chord length where there is boundary layer detachment. These simulations were also used to test appropriate airflow conditions, models, and meshes in the initial stages of the study, helping to inform their final iterations. However, the simulations produced results dissimilar to previous papers. The discrepancy is attributed to a lack of refinement in the mesh used in SimScale due to the limited resources available for this study. As a result, OpenFoam is used for subsequent simulations, where no limit on computational resources of the software is present.

The PIMPLE solver in PISO mode in OpenFOAM is used to perform a transient simulation with NACA0012 at a chord length of $0.10m$. An intensity of 0.15% is chosen to investigate laminar separation bubbles and to compare with previou sstudies. At the designated R_e, density of $1.0kg/m^3$ and kinematic viscosity of $1.50e-05m^2$/s, the angle of attack was varied by adjusting the x and y components of the free-stream velocity. A RAS simulation type with turbulence model $kw-sst$ is used.

The $k-\omega$ SST model was employed in this study due to its suitability for low R_e flows, combining the accuracy of $k-w$ modeling for near wall turbulence with the freestream stability of the k-e model for reduced sensitivity [13]. Three meshes are created and a grid convergence test is conducted. The chosen mesh averages $y+ < 3$ near the wall to ensure the viscous sub-layer is fully resolved.

Parameter	Mesh 1	Mesh 2	Mesh 3
Points	267408	309714	350816
Cells	132736	153750	174230
Max. Aspect Ratio	219.858	255.273	349.385
Max Non-Orthogonality	49.576	49.7978	50.8037
Max Skewness	0.719315	0.725961	0.674703
Average Lift Coefficient	0.796177	0.804454	0.812118
Average Drag Coefficient	0.078603	0.076696	0.066839

Table 1: Table detailing the parameters of the three constructed meshes.

The lift and drag coefficients of Mesh 1 and 2 shown in Table 1 has a difference of 1.03% and 2.43% respectively, whereas Mesh 2 and 3 have a difference of 0.95% and 12.85% for lift and drag coefficient respectively. The difference between Mesh 2 and 3 compared to Mesh 1 and 2 is attributed to the difference in max aspect ratio, max skewness, and max non-orthogonality. Due to the grid convergence results, Mesh 2 shown in Figure 2 was chosen to conduct the study.

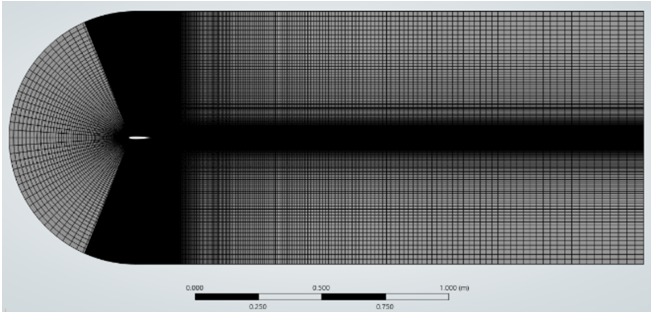


Figure 2: Mesh 2 in Ansys Meshing, with scale.

http://www.imavs.org/

The area around the airfoil shown in Figure 3 was further refined, however additional refinements of the mesh were not performed due to computational limitations caused by increasing file sizes and simulation run time from increasing vertical cell count.

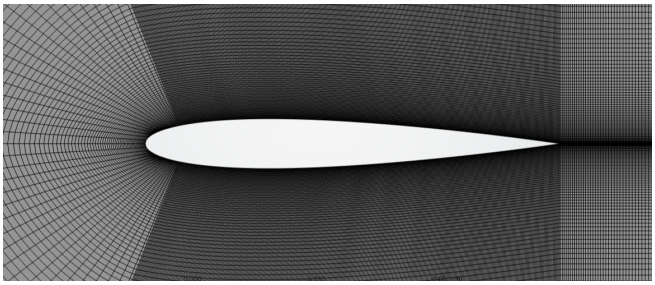


Figure 3: Mesh 2 near airfoil in Ansys Meshing.

The inlet for the boundary conditions has a fixed value condition while the topAndBottom and outlet uses inletOutlet. A no slip condition is applied to the airfoil. A two-second simulation without actuators at a maximum Courant number of one was conducted to validate boundary conditions. To implement the actuators, in the Navier-Stokes Equation 2, the momentum is applied as a body force F with dimensions in $\mathrm{m/s^2}$.

$$\frac{\partial \bar{u}}{\partial t} + \bar{u} \cdot \nabla \bar{u} = -\frac{1}{\rho}\nabla P + \nu \nabla^2 \bar{u} + F \tag{2}$$

Where $\bar{u}$ is the velocity vector in m/s, ρ is the density in kg/m^3, *P* is the pressure in Pa, ν is the kinematic viscosity in m^2/s, and *F* is the body force per unit mass in m/s^2.

To simulate the actuator, a rotatedBoxToCell cellset is created in topoSetDict to represent the plasma region on the airfoil and is positioned as discussed in Section 3. As demonstrated in the numerical study by Bernal Orozco et al. [13], using the Kloker model, the plasma region reaches up to $0.003m$ from the wall. Additionally, the force only extends significantly downstream the length of the embedded electrode. The region therefore has a length of $20mm$ and a height of $3mm$ [14]. The fvOptions utility is used to implement the plasma as a momentum source. The maximum body force is $416N/m^3$ as found by Hofkens [15] for a power source of $8kV$ and a frequency of $2kHz$. Maden et al [16] determined that the plasma distribution could be modeled as function of x and y coordinates. The model is calibrated to Hofkens' value [15] and integrated over the region to obtain the average force. The resulting integrated force density is determined to be $304.9N/m^3$.

The unsteadiness of the body force is then modeled using equation 3 [13].

$$f(x, y, t) = f(x, y)\sin^2(2\pi f t) \tag{3}$$

Using the vectorSemiImplicitSource option, a table with the time and its respective body force value is created for the duration of the simulation. The file is then read in the fvOptions and simulations are conducted with the actuators on and off.

All simulations write a .dat file containing the force coefficients and moments (lift, drag, pitching moment etc.). To simplify the processing of this data, a python script is written to output specific columns selected by the user in a csv file when a .dat file created by the simulation is given.

The meshes required for the study were created using Ansys meshing and exported as fluent .msh files where they were translated by OpenFOAM using the fluentMeshToFoam tool. All dimensions of the mesh were based on a single standard element size with all specific element sizes being a factor of the standard size. In the grid convergence, further refined meshes were made by reducing the standard element size to increase the total number of cells/elements by 10-15% per iteration.

To focus the refinement of the mesh around the airfoil a smooth transition bias is used with the growth rate along the height of the mesh being set to 1.04. This number is chosen as it is needed to keep the cells surrounding the airfoil edge in the wall function while optimizing the use of computational resources. The growth rate along the edge of the mesh behind the airfoil is set to 1.0175 to create a smooth gradient between the regions above and below the airfoil and the region behind the airfoil. The vertical bias used causes the cell aspect ratio to significantly increase for cells further vertically from the airfoil edge. However, due to limits in computational resources, further refinement of areas far from the airfoil edge vertically is not performed.

To improve simulations organization, a program is written using python that keeps track of simulation progress while redirecting the terminal output of the simulation into a file. The program reads the end time of the simulation from the controlDict file and converts it to a percentage displayed to the user along with an estimated completion time.

An assumption embedded into the OpenFOAM simulations is that the airfoil is perfectly smooth. This assumption differs from both experimental wind tunnel conditions and literature simulation values that employed the software XFoil. XFoil assumes low turbulence and relatively smooth surfaces [12]. In addition, the chord length in the Jardin et al. experiment is $0.15m$ whereas the following study is $0.10m$. Both factors are unlikely to have a significant impact on the results due to minimal differences, validated by the baseline lift coefficient values similarity to those in current studies.

To determine the location of flow separation at each simulated angle of attack, a Python script is written to process the results. The location of the separation point, identified by its distance along the chord of the airfoil from the leading edge, is approximated by searching for the point closest to the airfoil's leading edge with negative x-airflow velocity. The negative x-velocity signifies the presence of backflow, indicating appreciable separation of the boundary layer at that location.

http://www.imavs.org/

http://www.imavs.org/

This method did not determine exact separation points, but rather slightly further points where the boundary layer had fully separated, yielding an approximation of the true location.

The script can only process a single timestep of each simulation at a time, therefore the separation point for a given angle of attack is obtained from a single iteration of its simulation. Thus, the last iteration of each simulation is processed for this purpose. A visual analysis of several simulations indicates that this method is unlikely to produce significant error in the location of the separation points, as the leading side of the separation bubbles do not make significant position changes between timesteps.

3 Tests and Results

Figure 4 displays the position of flow separation, as defined in Section 2, as a function of angle of attack for the actuated and non-actuated airfoil. The results are consistent with the established understanding that as the angle of attack increases, the flow separation point approaches the leading edge. The flow separation points for angles of attack greater than $4°$, for the non-actuated airfoil, are within $20mm$ of the airfoil's leading edge. Since the length of the actuator used is $20mm$, and DBD plasma actuators are most effective when positioned just before the separation point [17] , it is concluded that the optimal position of the actuator is with its front edge at the leading edge of the airfoil.

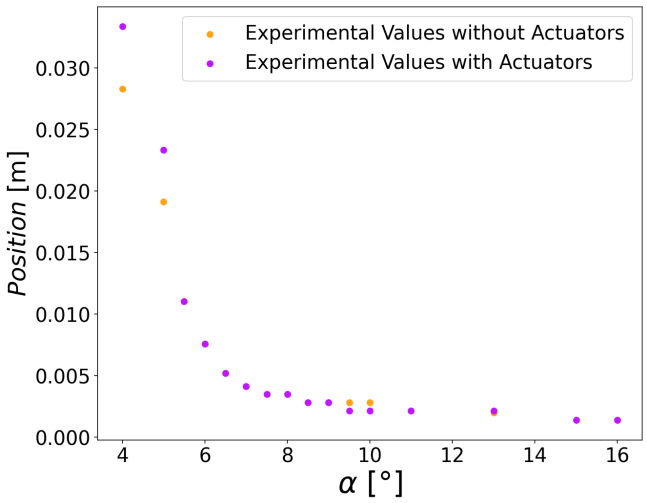


Figure 4: Graph of simulated position of flow separation vs angle of attack for actuated and non-actuated NACA0012 airfoil.

As shown in Figure 4, the majority of the positions of flow separation from the actuated simulations are at the same location as their corresponding non-actuated simulation, though some positions were further from the leading edge when actuated, such as $5°$ as shown in Figure 5. There are two exceptions, those being $9.5°$ and $10°$ which have flow separation closer to the leading edge when actuated. These exceptions are attributed to errors within the simulations, which may be amplified by the mesh refinement selected and discussed in Section 2.

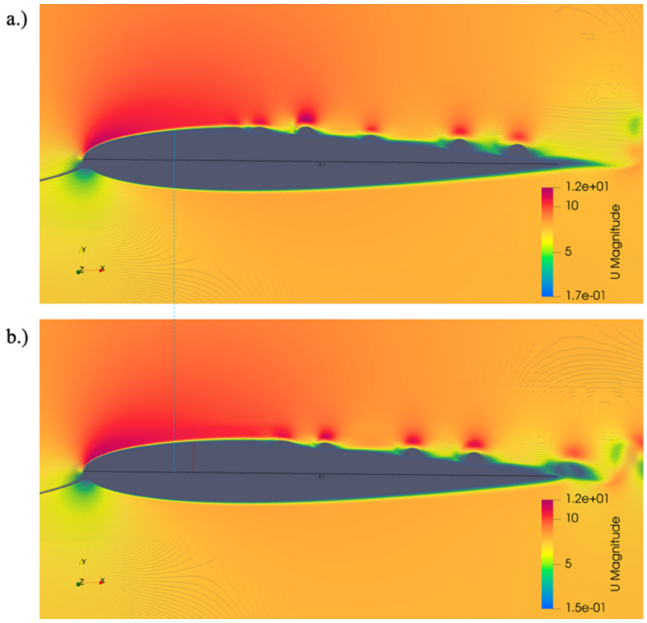


Figure 5: Airfoil at $5°$ angle of attack showing flow separation. Actuators delayed flow separation from 1.91 cm (a) to 2.33 cm (b).

Simulations with and without the actuator are conducted to validate its effectiveness at this position. Figure 6 compares the actuated and non-actuated lift values from these simulations to the non-actuated values found by Jardin et al [12].

The results demonstrate the actuator's effectiveness at increasing lift when placed at this location, as all simulated angles, with the exceptions of $9°$ and $15°$, yielded higher time - averaged lift coefficients when actuated. These exceptions are attributed to errors within the simulations.

Values from the Jardin study display that the stall angle for $R_e = 53,000$ is approximately $10°$. However, although the simulated data initially follows a similar trend, it does not display a stall angle within the experimental range. All results after $10°$ should therefore be interpreted cautiously. This overprediction can be attributed to a variety of reasons which Tank et Al [18] explored in depth. Notably, laminar separation and reattachment, unique to low R_e, is difficult to capture. Other research including Alleto's dynamic stall analysis [19] suggest that the Langtry-Menter $kw - sst$ model, also known as the $\gamma - \mathrm{R}_{e\theta}$ model, is more suitable for turbulence transition modeling. This should be explored in future works.

The lack of stall could also be attributed to the methodology. This study utilized time-averaged lift coefficients, which

http://www.imavs.org/

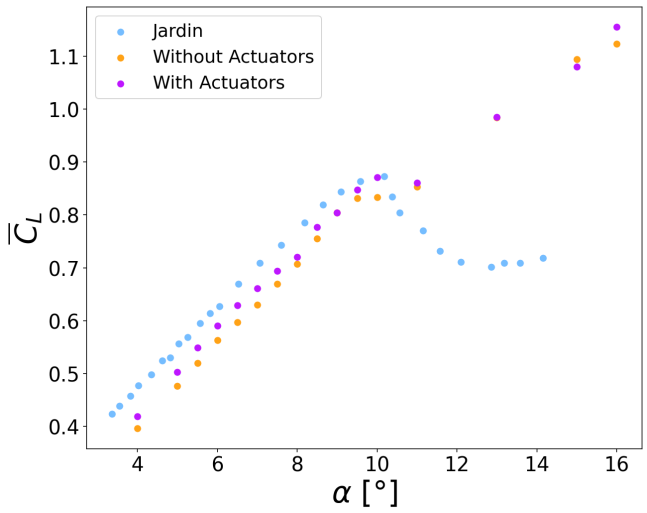


Figure 6: Graph of simulated lift coefficient vs angle of attack for non-actuated and actuated NACA0012 airfoil with comparison to Jardin literature values without actuators.

may not have fully captured transient stall conditions. At higher angles of attack around the anticipated stall angle of $10°$, the lift coefficient exhibited increased fluctuations compared to lower angles. These fluctuations increase until $16°$. Figure 7 shows the amplitude of these fluctuations at $5°$, $8.5°$, and $10°$ significantly increasing with angle of attack, indicative of stall behavior.

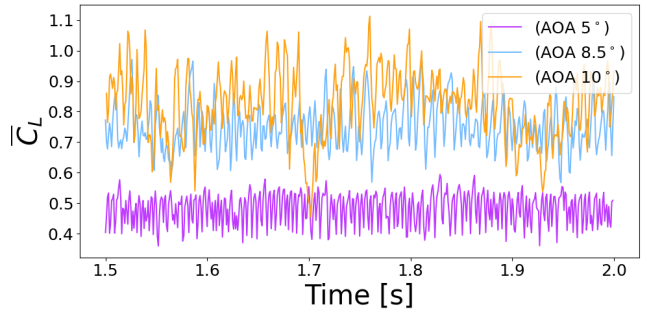


Figure 7: Lift coefficient from $1.5-2.0s$ for $5°$, $8.5°$, and $10°$ angle of attack with no actuators showing increasing fluctuations indicative of flow instability.

The sudden decrease in lift coefficient is attributed to laminar separation bubbles bursting [20], as shown in Figure 8. Despite these pronounced oscillations, the time-averaged lift coefficient continues to show an overall increasing trend.

Additionally, the $y+$ is greater than one due to limited computational resources. To ensure accuracy while fully resolving the viscous sub – layer, future works should further refine the mesh near the wall to $y+ < 1$.

Figure 9 displays the simulated coefficient of drag vs. angle of attack for the actuated and non-actuated simulations. The plot confirms the actuator's effectiveness at decreasing drag when placed at the airfoil's leading edge, as all simulated angles, with the exceptions of $11°$ and $13°$, yielded lower drag coefficients when actuated. These exceptions are attributed to errors within the simulations.

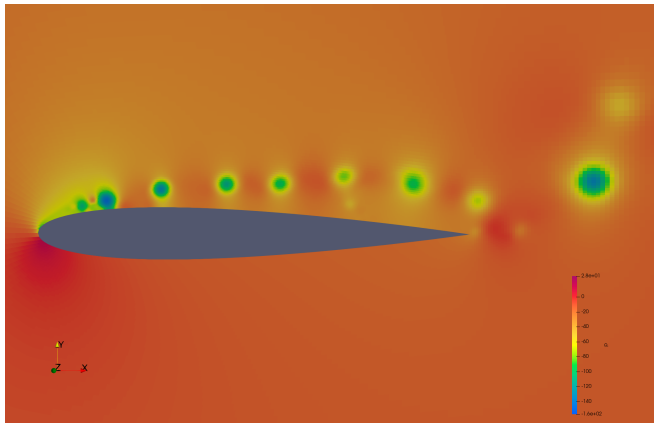


Figure 8: Laminar separation bubbles at $10°$ angle of attack with no actuator.

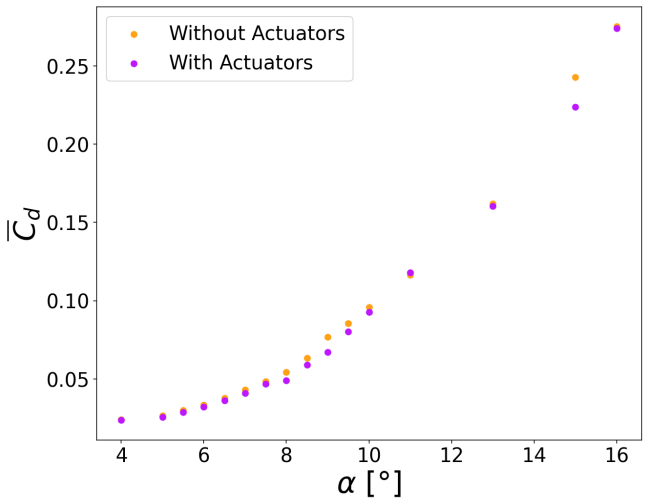


Figure 9: Graph of simulated drag coefficient vs angle of attack for non-actuated and actuated NACA0012 airfoil.

4 Conclusion

This study set out to find the optimal positions for DBD plasma actuators to mitigate flow separation and increase the stall angle for flight at low R_e. This technology has applications in improving flight performance in MAVs through expanding the range of conditions these vehicles can be expected to operate in.

The locations of flow separation for angles of attack past $5°$ are within the actuator length of $20mm$. Thus, the opti-

mal location of the actuator is directly at the leading edge of the airfoil. This location is validated by simulations with the actuator at this position. Most actuated simulations did not demonstrate an ability of the actuator to delay flow separation to further along the chord length. However, the actuated simulations display a trend of yielding higher lift and lower drag coefficients than the corresponding non-actuated simulations, demonstrating the effectiveness of the actuator at this location.

This study was not able to investigate the actuator's ability to increase the stall angle. Neither the actuated nor non-actuated simulations demonstrated a stall angle within the experimental range, which did not agree with data from previous studies used for comparison due the turbulence model used and the near-wall mesh refinement. Thus, the stall angle delay should be further explored in future studies.

Fabrication of a DBD plasma actuator for future investigation would include a thin electrode, possibly made of or containing graphene derivatives or electroplated silver. This would be placed on a dielectric material comprised of Kapton due to its low cost, flexibility, and high dielectric strength of 275.6 $V/\mu m$ at a thickness of 25.4 μm [21].

The actuator should implement a separate battery power source, as it is not ideal to interfere with the main MAV power. It can be built by using a MOSFET with a tailored threshold to control the frequency ($2kHz$), a transformer that can step up battery voltage into the ideal $8kV$ amplitude, a system (i.e. optocoupler) that can isolate the high voltage from the low, and resistors to ensure one meets the required voltage.

http://www.imavs.org/

ACKNOWLEDGMENTS

This work is part of the Queen's University Aerospace Design Team which received funding from the Smith Faculty of Engineering and Applied Sciences. The authors thank Sam Williamson, Dr. Francesco Ambrogi, Dr. Barbara L. da Silva, and Dr. John Kurelek for their contributions. This work was also supported by SimScale through their Academic Plan, which was used for preliminary simulations.

REFERENCES

[1] H. Sturm, G. Dumstorff, P. Busche, and D. Westermann. Boundary layer separation and reattachment detection on airfoils by thermal flow sensors. *Sensors*, 12(11):14292–14306, November 2012.

[2] J. Winslow, H. Otsuka, B. Govindarajan, and I. Chopra. Basic understanding of airfoil characteristics at low reynolds numbers (10–10). *Journal of Aircraft*, 55(3):1050–1061, 2018.

[3] R. Dumas, C. Renaud, and M. Villedieu. Numerical predictions of low reynolds number compressible aerodynamics. *Aerospace Science and Technology*, 85:211–223, 2019.

[4] N. Thipyopas and J.-M. Moschetta. A fixed-wing biplane mav for low speed missions. *International Journal of Micro Air Vehicles*, 1(1):13–24, 2009.

[5] S. Zhong, X. Wang, and Y. Liu. Control of laminar flow separation over a backward-facing rounded ramp with c-d riblets – the effects of riblet height, spacing and yaw angle. *International Journal of Heat and Fluid Flow*, 81:108629, 2020.

[6] A. Heidarian, H. Ghassemi, and P. Liu. Numerical analysis of the effects of riblets on drag reduction of a flat plate. *Journal of Applied Fluid Mechanics*, 11(3):679–688, 2018.

[7] F. Rodrigues, J. Páscoa, and F. Dias. Power consumption characterization of dbd plasma actuators for boundary layer control. In *International Conference on Engineering*, Covilhã, Portugal, 2015.

[8] R.-B. Liu, W.-T. Wei, H.-P. Wan, and J.-Y. Cheng. Experimental study on airfoil flow separation control via an air-supplement plasma synthetic jet. *Advances in Aerodynamics*, 4(1):35, 2022.

[9] L. Francioso, C. De Pascali, E. Pescini, M. G. De Giorgi, and P. Siciliano. Modeling, fabrication and plasma actuator coupling of flexible pressure sensors for flow separation detection and control in aeronautical applications. *Journal of Physics D: Applied Physics*, 49(23):235201, 2016.

[10] R. Pereira, G. J. W. van Bussel, and W. A. Timmer. Active stall control for large offshore horizontal axis wind turbines; a conceptual study. *Journal of Physics: Conference Series*, 555(1):012082, 2014.

[11] K. Asai, K. Nakano, and Y. Nonomura. Stalling characteristics of airfoils near the boundary between leading-edge stall and trailing-edge stall. *AIAA Journal*, 58(10):4291–4303, 2020.

[12] T. Jardin, V. Ferrand, and E. R. Gowree. Naca0012 airfoil at reynolds numbers between 50,000 and 140,000 — part 1: Steady freestream. *International Journal of Heat and Fluid Flow*, 112, 2025.

[13] R. A. Bernal Orozco, O. M. Huerta Chávez, J. Á. Ortega Herrera, and A. Arias Montaño. Numerical technique for implementation of sdbd plasma actuators. *Computación y Sistemas*, 26(4):1527–1538, 2022.

[14] M. B. S. Forte, F. Jolibois, M. P. Mignolet, E. Moreau, G. Artana, and G. Touchard. Optimization of a dielectric barrier discharge actuator by stationary and non-stationary measurements. *Experiments in Fluids*, 43(6):917–928, 2007.

[15] A. Hofkens. Determination of the body force generated by a plasma actuator through numerical optimization. Master's thesis, Delft University of Technology, 2016.

[16] I. Maden, A. Aksu, A. Topal, H. Yildiz, and E. Can Yilmaz. Experimental and computational study of the flow induced by a plasma actuator. *International Journal of Heat and Fluid Flow*, 41:80–89, 2013.

[17] T. C. Corke, P. O. Bowles, C. He, and E. H. Matlis. Sensing and control of flow separation using plasma actuators. *Philosophical Transactions of the Royal Society A*, 369(1940):1459–1475, 2011.

[18] J. Tank, L. Smith, and G. R. Spedding. On the possibility (or lack thereof) of agreement between experiment and computation of flows over wings at moderate reynolds number. *Interface Focus*, 7(1):20160076, 2017.

[19] M. Alletto. Dynamic stall naca0012. `https://wiki.openfoam.com/index.php?title=Dynamic_stall_NACA0012_Michael_Alletto&oldid=4388`, 2023. OpenFOAM Wiki.

[20] M. Ahmad N. Ali and R. R. Raja. Separation bubble characteristics over a naca 2415 airfoil at low reynolds number. *Journal of Physics: Conference Series*, 2054(1):012005, 2021.

[21] Jet Propulsion Laboratory. Material notes and tables. `https://descanso.jpl.nasa.gov/SciTechBook/series3/07Chapter6MatlNotesTables.pdf`, 2011. Chapter 6, SciTech Book Series III, Pasadena, CA.

http://www.imavs.org/

Bibliography

[1] Nilda Xolo-Tlapanco and Jose Martinez-Carranza. 3d cnn and multi-head attention to relative camera position. In Jose Martinez-Carranza, editor, *16th International Micro Air Vehicle Conference and Competition*, pages 18–24, San Andrés Cholula, Puebla, Mexico, Nov 2025. Paper no. IMAV2025-1. 14

[2] Brianna Balam-Velasco, L. Oyuki Rojas-Perez, and Jose Martinez-Carranza. A review of visual slam systems and single-image depth estimation for challenging scenarios. In Jose Martinez-Carranza, editor, *16th International Micro Air Vehicle Conference and Competition*, pages 25–31, San Andrés Cholula, Puebla, Mexico, Nov 2025. Paper no. IMAV2025-2. 14

[3] Guilherme Gabriel, Bruno Sarmento, and Laisa Costa. Synthetic dataset generation for object detection in mav applications. In Jose Martinez-Carranza, editor, *16th International Micro Air Vehicle Conference and Competition*, pages 32–36, San Andrés Cholula, Puebla, Mexico, Nov 2025. Paper no. IMAV2025-3. 14

[4] Pedro Becerra, Blanca Briseño, Diego Alexis Martínez, and Hugo Rodríguez. Intelligent vision-based quadrotor path follower. In Jose Martinez-Carranza, editor, *16th International Micro Air Vehicle Conference and Competition*, pages 37–42, San Andrés Cholula, Puebla, Mexico, Nov 2025. Paper no. IMAV2025-4. 14

[5] Fabián González-Tejeda, Brenda Ivette García-Maya, and Hernan Abaunza Gonzalez. Autonomous uav landing with visual servoing and semi-markov decision processes. In Jose Martinez-Carranza, editor, *16th International Micro Air Vehicle Conference and Competition*, pages 43–50, San Andrés Cholula, Puebla, Mexico, Nov 2025. Paper no. IMAV2025-5. 14

[6] Zhanghao Lu, Simon Watson, and Andrew Weightman. Model-based analysis of uav accurate landing in stochastic turbulent environments. In Jose Martinez-Carranza, editor, *16th International Micro Air Vehicle Conference and Competition*, pages 51–58, San Andrés Cholula, Puebla, Mexico, Nov 2025. Paper no. IMAV2025-6. 14

[7] Cansu Yıkılmaz and Christophe De Wagter. Reinforcement learning-based optimal guidance for landing the variable skew quad plane on a ship. In Jose Martinez-Carranza, editor, *16th International Micro Air Vehicle Conference and Competition*, pages 59–68, San Andrés Cholula, Puebla, Mexico, Nov 2025. Paper no. IMAV2025-7. 14

[8] Yuki Onda, Abner Asignacion, , and Satoshi Suzuki. A lidar-based deep reinforcement learning autonomous navigation system in unknown environments. In Jose Martinez-Carranza, editor, *16th International Micro Air Vehicle Conference and Competition*, pages 69–75, San Andrés Cholula, Puebla, Mexico, Nov 2025. Paper no. IMAV2025-8. 14

[9] Daniel Aláez, Manuel Prieto, Jesús Villadangos, and José Javier Astrain. Radio coverage-aware uav planning and navigation in urban scenarios. In Jose Martinez-Carranza, editor, *16th International Micro Air Vehicle Conference and Competition*, pages 76–81, San Andrés Cholula, Puebla, Mexico, Nov 2025. Paper no. IMAV2025-9. 14

[10] Carlos Osorio Quero and Jose Martinez-Carranza. Adrc-driven trajectory control of a quadcopter considering ground effect dynamics. In Jose Martinez-Carranza, editor, *16th International Micro Air Vehicle Conference and Competition*, pages 82–89, San Andrés Cholula, Puebla, Mexico, Nov 2025. Paper no. IMAV2025-10. 14

[11] Jialiang Wang, Yijun Huang, Yizhou Chen, Zongzhou Wu, Qigeng Wu, Zhiyu Zhou, Xi Chen, Zhi Gao, and Ben M. Chen. Cu-fundamental: A comprehensive simulation framework for autonomous mav. In Jose Martinez-Carranza, editor, *16th International Micro Air Vehicle Conference and Competition*, pages 90–97, San Andrés Cholula, Puebla, Mexico, Nov 2025. Paper no. IMAV2025-11. 14

[12] Francisco Anguita Chamorro, Rafael Perez Segui, Carmen De Rojas, Miguel Fernandez Cortizas, Javier Melero Deza, and Pascual Campoy. A comparative study on thrust map estimation for multirotor aerial vehicles. In Jose Martinez-Carranza, editor, *16th International Micro Air Vehicle Conference and Competition*, pages 98–105, San Andrés Cholula, Puebla, Mexico, Nov 2025. Paper no. IMAV2025-12. 14

http://www.imavs.org/

[13] Karl Stol and Salim Al-zubaidi. A comparison of optimised, fixed-tilt, fully-actuated multirotor uav configurations. In Jose Martinez-Carranza, editor, *16th International Micro Air Vehicle Conference and Competition*, pages 106–111, San Andrés Cholula, Puebla, Mexico, Nov 2025. Paper no. IMAV2025-13. 14

[14] Carlos Perez-Rodriguez, Cesar Martinez-Torres, and Jose Martinez-Carranza. A framework for indoor-monocular depth estimation on uavs using a thermal camera. In Jose Martinez-Carranza, editor, *16th International Micro Air Vehicle Conference and Competition*, pages 112–117, San Andrés Cholula, Puebla, Mexico, Nov 2025. Paper no. IMAV2025-14. 14

[15] Jesus Rodriguez, Julio Sebastián De la Trinidad-Rendón, Jose Martinez-Carranza, and Hugo Gustavo González-Hernández. Energy-aware hybrid event-triggered control for micro-drones. In Jose Martinez-Carranza, editor, *16th International Micro Air Vehicle Conference and Competition*, pages 118–124, San Andrés Cholula, Puebla, Mexico, Nov 2025. Paper no. IMAV2025-15. 14

[16] Yuki Ohira, Abner Asignacion, and Satoshi Suzuki. Payload state estimation for cooperative manipulation using multiple uavs. In Jose Martinez-Carranza, editor, *16th International Micro Air Vehicle Conference and Competition*, pages 125–132, San Andrés Cholula, Puebla, Mexico, Nov 2025. Paper no. IMAV2025-16. 14

[17] Michael van Breukelen Castillo, Christophe De Wagter, Robin Ferede, Reinier Vos, and Guido de Croon. Spiking neural networks for high-speed continuous quadcopter control using proximal policy optimization. In Jose Martinez-Carranza, editor, *16th International Micro Air Vehicle Conference and Competition*, pages 133–141, San Andrés Cholula, Puebla, Mexico, Nov 2025. Paper no. IMAV2025-17. 14

[18] Tomoyuki Matsuda, Saad Hussain, Abner Asignacion, and Satoshi Suzuki. Design of frontal perching mechanism to vertical surfaces for flapping-wing mavs. In Jose Martinez-Carranza, editor, *16th International Micro Air Vehicle Conference and Competition*, pages 142–147, San Andrés Cholula, Puebla, Mexico, Nov 2025. Paper no. IMAV2025-18. 14

[19] Saad Hussain, Abner Asignacion, and Satoshi Suzuki. Pecking mechanism inspired by avian cranial kinesis for flapping-wing aerial manipulators. In Jose Martinez-Carranza, editor, *16th International Micro Air Vehicle Conference and Competition*, pages 148–155, San Andrés Cholula, Puebla, Mexico, Nov 2025. Paper no. IMAV2025-19. 14

[20] Maya Pivert, Sacha De Sousa, Renaud Kiefer, Sylvain Durand, and Edouard Laroche. Flight pattern and experimental flights of a quadricopter drone at the edge of a smoke plume. In Jose Martinez-Carranza, editor, *16th International Micro Air Vehicle Conference and Competition*, pages 156–161, San Andrés Cholula, Puebla, Mexico, Nov 2025. Paper no. IMAV2025-20. 14

[21] Robin Krasinkiewicz, Maggie Gaskell, Noah Maggiani, Janet Liu, Haowen Yan, Charlie Smith, Annie Wu, and Aidan Woods. Dielectric barrier discharge plasma actuators to mitigate flow separation. In Jose Martinez-Carranza, editor, *16th International Micro Air Vehicle Conference and Competition*, pages 162–168, San Andrés Cholula, Puebla, Mexico, Nov 2025. Paper no. IMAV2025-21. 14

http://www.imavs.org/