12th International Micro Air Vehicle Conference

November 17-19, 2021

Puebla, México

www.imavs.org

Jose Martinez-Carranza
Editor

Contents

Preface

On behalf of the Local Organising Committee, It is my pleasure to present the proceedings of the 12th International Micro Air Vehicle Conference, which was held in Puebla, México from November 17-19, 2021. 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.

For the first time ever, The 12th International Micro Air Vehicle Conference was run by Latin American academic institutions based in Mexico: Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), Universidad de las Américas Puebla (UDLAP), Benemérita Universidad Autónoma de Puebla (BUAP) and Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional (CINVESTAV), Unidad Zacatenco.

The IMAV is a pioneer scientific-technological event in the field of aerial robotics and has been established as a stellar event among the communities of researchers dedicated to the study, development and research of Micro Air Vehicles. This year of 2021, due to the COVID-19 pandemic, the IMAV-2021 was run in a virtual mode as a conference only; this is, no competitions were run this time. Yet, I believe the event offered a huge opportunity to communicate the latest developments regarding Micro Air Vehicles as much as to foster collaborations among the members of the IMAV international community.

These proceedings contain 27 peer-reviewed scientific papers by 88 authors organised in 9 sessions presented at the IMAV-2021. 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 on the field of Micro Air Vehicles. In the awards ceremony of the conference, the following awards were announced: Best Conference Paper, Best Technical/Application Paper, and Best Student Paper. Based on the quality of the scientific and technical contribution, some papers were selected to be published in two scientific journals: the International Journal of Micro Air Vehicles (Sage), and Unmanned Systems (World Scientific).

In addition to the presentation of the scientific papers, 6 keynote talks were delivered by experts in the field. The first speaker, Rear Adm Leopoldo Díaz, Head of the Research and Technological Development Unit of the Mexican Navy, presented the research and technological work on UAS that has been developed in this Unit. Prof. Guido de Croon from TUDelft, talked about his work on insect-inspired AI for swarms and tiny drones. Prof. Tom Richardson from the University of Bristol, talked about studying volcanic emissions by operating drones beyond visual line of sight. Brandon Gilles, CEO of Luxonis, introduced the OAK-D Lite smart camera capable of running Spatial AI on the chip and its use on drone applications. Nicolas Marchand from GIPSA-lab/CNRS, presented his work on event-based control for flying robotics. Finally, Rogelio Lozano from the Université de Technologie de Compiègne, presented his work on the dynamical model of a mini helicopter without a swashplate and the challenges behind this problem in outdoor flight.

We also had the privilege of gathering former IMAV General Chairs who shared nice memories of previous IMAVs. The participation of these former Chairs was split into 2 panel sessions. The first one was attended by Prof. Simon Watkins from RMIT and Prof. Ben M. Chen from CUHK, Chairs of the IMAV 2016 and IMAV 2018, respectively. The second panel was attended by Prof. Guido de Croon from TUDelft, Prof. Jean-Marc Moschetta from ISAE, Dr. Gautier Hattenberger from ENAC, and Prof. Pascual Campoy from UPM, Chairs of IMAV 2014, 2017 and 2019. I have no doubt these sessions summarised the influence and legacy that IMAV conferences and competitions have had in the field of Micro Air Vehicles over the last years.

Finally, my deepest gratitude goes to all members of the Local Organising Committee for their invaluable support in the organisation of this IMAV-2021, even during this difficult pandemic period. Also, I appreciate the guidance and support of the members of the International Committee, whose enthusiasm and kindness inspired us to do our best to prepare and run this event. Last but not least, we are very grateful for the sponsorship of the awards provided by LUXonis.

Puebla, México. November 2021

Jose Martinez-Carranza
Instituto Nacional de Astrofísica, Óptica y Electrónica

International IMAV Committee

Adrián Carrio	UPM, Madrid, Spain
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
Florian Holzapfel	TUM, München, Germany
Gautier Hattenberger	ENAC, Toulouse, France
Guido de Croon	TUDelft, Delft, The Netherlands
Hector J. G. de Marina Peinado	UCM, Madrid, Spain
Henry De Plinval	ONERA, Toulouse, France
Jean-Marc Moschetta	ISAE, Toulouse, France
Jose Martinez-Carranza	INAOE, Puebla, Mexico
Mark Reeder	AFIT, Ohio, USA
Mohamed Abdulghani	RMIT, Melbourne, Australia
Paloma Puente	UPM, Madrid, Spain
Pascal Morin	UPMC, Toulouse, France
Pascual Campoy	UPM, Madrid, Spain
Sergey Shkarayev	UA, Tucson, USA
Shupeng Lai	NUS, Singapore, Singapore
Simon Watkins	RMIT, Melbourne, Australia

IMAV2021 Local Organising Committee

Jose Martinez-Carranza	General Chair
Hugo Rodríguez-Cortés	Program Chair
Cesar Martínez-Torres	Deputy Chair
José Fermi Guerrero-Castellanos	Deputy Chair
Antonio Juárez-González	Website Chair
Leticia Oyuki Rojas-Perez	Logistics
Israel Cruz-Vega	Session Chair
Caleb Rascon	Session Chair
Aldrich Alfredo Cabrera-Ponce	Support Staff
Jose Arturo Cocoma-Ortega	Support Staff
Antonio Matus-Vargas	Support Staff
Aaron Lopez-Luna	Support Staff
Sergio Serrano	Support Staff
Manuel Lopez Garcíca	Support Staff
Juan Pablo Romero-Flores	Support Staff
Esteban Jimérez-Ruiz	Support Staff
Emmanuel López-Pérez	Support Staff
José Juan Legaria-Perramón	Support Staff
René Parlange Chavarría	Support Staff

Sponsors

Awards Sponsor

LUXonis

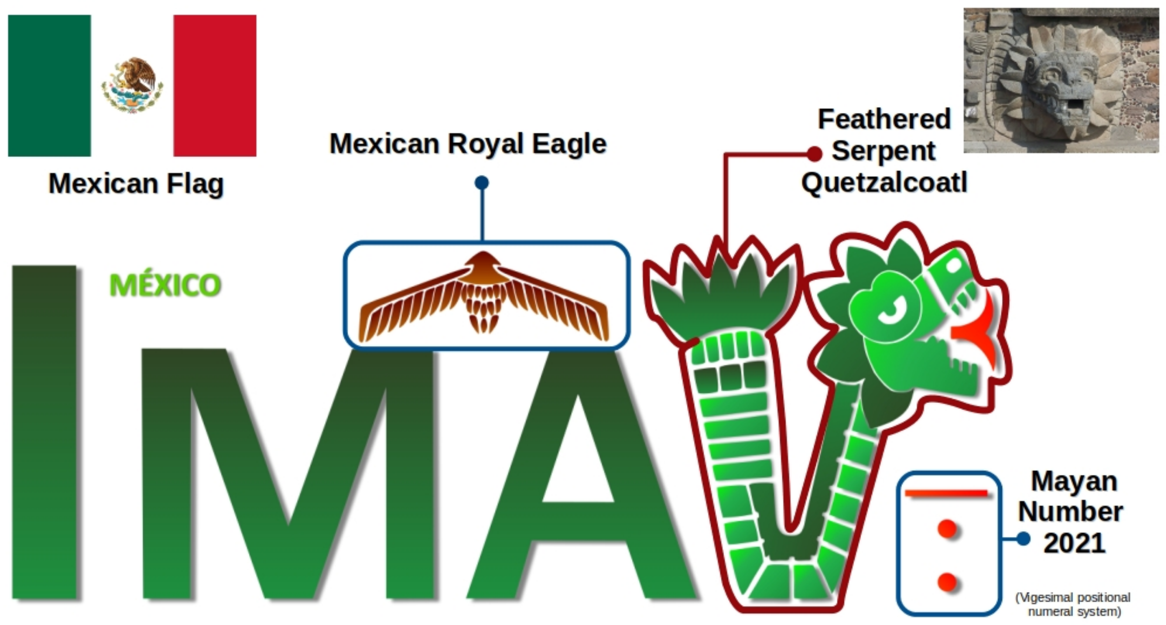
See more. Do more.

https://www.luxonis.com/

Meaning of the IMAV 2021 Logo

The logo of the IMAV 2021 was inspired by the Mexican flag, seen at the top left of the image below. The logo shows the letters of the acronym IMAV in green, the first colour of the flag. At the top of the log we see a figure with a fixed-wing like shape. This is intended to represent a fixed-wing MAV dressed with brown feathers, representing the *Mexican Royal Eagle*, located at the center of the Mexican flag. The letter ’V’ in the acronym has been depicted in the shape of a snake. This symbol represents a Deity from the Aztec culture: the *feathered Serpent* popularly known as *Quetzalcoatl*. The tongue of the snake is in red as in the Mexican flag, and also located to the right. Finally, a bar and two dots in red are seen at the right bottom part of the logo, representing a Mayan number, a vigesimal positional numeral system that uses bars and dots to represent numbers. In this logo, the Mayan number represents the number 2021. The logo also has the word ”México” in a smaller font size, above the acronym. Therefore, buy putting everything together and with some imagination, the logo portraits a Mexican flag shaped by the IMAV acronym, the year of the conference and containing representative symbols of the Mexican culture.

We are thankful to Guadalupe Cabrera-Ponce for her design on the Quetzalcoatl symbol and to Oyuki Rojas-Perez for her design on the eagle and Mayan number symbols. Their help was essential to create this IMAV 2021 logo. Jose Martinez-Carranza conceptualise and supervised the elaboration of the logo.


Mexican Flag
Mexican Royal Eagle
Feathered Serpent Quetzalcoatl
MÉXICO
IMAV
Mayan Number 2021
(Vigesimal positional numeral system)

Call for Papers

We cordially invite all experts, users, scientists, young researchers, and students being active or interested in the field of Micro Unmanned Aerial Vehicles to attend the 12th International Micro Air Vehicle Conference. This edition of the IMAV will be organized in the virtual model. The conference will take place November 17-19 2021. 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
- The 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 (note that the Article Processing Charges will be waived):

- International Journal of Micro Air Vehicles
- Unmanned Systems

Important Dates:

- *Paper Submission Deadline: June 21, 2021*
- *Notification of Acceptance: August 1st, 2021*
- *Camera-ready: August 30, 2021*
- *Registration: Before September 1, 2021*

The proceedings of the conference will be available on-line with free access for anybody.

For further questions, please visit the conference web pages http://imav2021.inaoep.mx/ http://www.imavs.org/

Best regards,

Hugo Rodríguez Cortés - Program Committee Chair
Jose Martinez-Carranza - General Chair

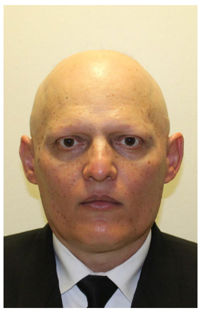
Conference Oral Presentations

Keynote Speakers

Urban Air Mobility, a vision by Airbus

Rear Adm. Leopoldo Jesus Diaz González Solórzano – Head of the Research and Technological Development Unit Mexican Navy

Biography



Admiral González Solórzano graduated as Naval Science Engineer from the Heroica Escuela Naval Militar of the Mexican Navy. He has held several appointments in the Mexican Navy such as Patrol commander, Deputy and later Area Head of the Deputy General Management Office of Communications and Informatics, Inspector of the Naval Command of the 3rd Naval Zone, and Permanent alternate Representative of Mexico to the International Maritime Organization, a specialised agency of the United Nations with Headquarters in London, United Kingdom. In 2016, he was promoted to Rear Admiral of the Mexican Navy. He has also received several distinctions throughout his career. He was awarded the naval merit of First-Class due to this high performance during his studies at the Heroica Escuela Naval Militar, and awarded the Argentine Navy Prize for his high performance in military skills during these studies. He received an Honorary Mention for his high performance during a course given by the international Hydrographic Organization. To date, he is the Head of the Research and Technological Development Unit of the Mexican Navy.

Abstract

Unmanned Aerial Vehicles developed by the Mexican Navy

The Research and Technological Development Unit is a department within the Mexican Navy responsible for developing projects that support the navy forces, units and naval establishments. The main research projects focus on Unmanned Aerial Systems (UAS), radars, datalink systems, sonars, simulators, and rockets. This talk will present the research, and technological work carried out since 2011 on UAS. To date, two aerial vehicles have been developed and tested in different operation environments; one of these vehicles is under the production of more units. The development of these vehicles implies a continuous learning curve, but in particular, it is essential to design adequate schemes for validation and verification of the vehicle's performance. Nowadays, with the support of the National Council of Science and Technology of Mexico, an Unmanned Aerial Vehicle (UAV) is under development, capable of performing vertical take-off and landing. Since two years ago, this vehicle has been in the stage of testing to validate its design and functionality. For future work, we are considering the deployment of UAS from naval ships, the development of cooperative UAVs with swarm capabilities and the use of alternative energies.

Autonomous Drone Racing

Prof. Guido de Croon, Full Professor in Bio-inspired Micro Air Vehicles Department of Aerospace Engineering Delft University of Technology

Biography

Received his M.Sc. and Ph.D. in the field of Artificial Intelligence (AI) at Maastricht University, the Netherlands. His research interest lies with 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 light-weight 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 Full Professor at TU Delft and scientific lead of the Micro Air Vehicle lab (MAV-lab) of Delft University of Technology.

Abstract

Insect-inspired AI for swarms of tiny autonomous drones

Swarms of tiny autonomous drones can help humans in search-and-rescue missions, in keeping track of the stock in warehouses or in monitoring crop in greenhouses. Tiny drones (think below 30 grams) are very safe for humans, are suitable for flying in narrow environments, and are very cheap so that they can be produced in large numbers. However, it is also notoriously hard to make such tiny drones fly autonomously. Due to their extremely low weight, they are very limited in terms of energy and payload. This implies that they can carry few sensors and extremely little processing compared to, e.g., self-driving cars. In my talk, I will talk about the effort at TU Delft's MAVLab to make swarms of tiny autonomous drones, and how we approach this by drawing inspiration from insect intelligence. I will present our work on the lightest autonomous drone in the world, the 20 gram "DelFly Explorer" and on a swarm of 33-gram Crazyflies that is able to autonomously explore an unknown environment and come back to the departure point. Furthermore, I will discuss our recent study in which we designed a swarm of CrazyFlie drones able to autonomously localize gas sources. I will end with our efforts into incorporating spiking neural networks in neuromorphic hardware on our drones, showing results of an evolved spiking neural controller that was successfully ported to the real world, for the first time controlling a flying drone with neuromorphic processing in the control loop.

Tom Richardson - Senior Lecturer in Flight Dynamics and Control Flight Lab University of Bristol

Biography

Tom is a senior lecturer in flight dynamics and control at the University of Bristol. With a PhD in nonlinear control system design, he specializes in the application of modern control theory and novel sensors to Unmanned Air Systems (UAS). Tom has held an NPPL (pilots license) for over 15 years, runs the University of Bristol glider flight test course, and has been responsible for UAS operations in multiple countries. He has been granted permission multiple times for Beyond Visual Line Of Sight (BVLOS) operations, and holds the University of Bristol CAA UAS Permission for Commercial Operations. He has also run flight demonstrations and test flights for DSTL, BAE Systems, QinetiQ, DSTL and Roke Manor. Tom is also a founding partner of Perceptual Robotics which has recently been awarded 'Robotics & AI in Extreme Environments' funding by Innovate UK for Offshore Wind Turbine Inspection. https://www.perceptual-robotics.com/

Abstract

Remote Sampling and Measurement of Volcanic Emissions using Drones

Recent drone developments are having a significant impact on the way that volcanic emissions are being studied. This talk will cover collaborations between the Bristol University Flight Lab and Earth Science colleagues on field campaigns to a range of volcanoes worldwide. Beyond Visual Line of Sight (BVLOS) operations have enabled ash samples to be collected and gas measurements to be made at distances up to 14km and altitudes up to 14,000ft above take-off. Target volcanoes include Fuego which is an active stratovolcano in Guatemala and is almost constantly active at a low level. Small gas and ash eruptions occur every 15 to 20 minutes and multiple flights have been carried out to collect a range of ash samples from within the plume. Most recently, Dr Richardson has been part of the international collaborative multi-drone 'ABOVE' field campaign to Manam and Rabul volcanoes in Papua New Guinea - the objective of which was to achieve the first simultaneous inter-comparison of ground, aerial, and satellite-based measurement techniques for volcanic gas (SO2) emissions.

Brandon Gilles - CEO Luxonis

Biography

Brandon is driven by the singular belief that the biggest impact he can have on the world is fostering innovation. And the understanding that a 5-fold increase in productivity is the difference between the middle ages and now. And the driver of that productivity increase - and the tremendous increase in the quality of life we all have - is innovation. And for innovation to happen, simplification needs to happen first. So that powerful things can be used easily and readily - and then combined with other powerful things. Brandon's mission is to make embedding performant, spatial AI and CV into products simple - to enable and foster a wave of innovation powered by being able to embed human-like perception into products across all sorts of industries.

Abstract

Spatial AI Meets Embedded Systems

The combination of high-resolution depth perception, real-time artificial intelligence, advanced computer vision functions, and high-frame-rate/high-resolution/multi-sensor cameras systems used to be only available to those with huge budgets. Monetary, size, weight, and power budgets. Now, it's possible to have all of this on an embedded system - in a tiny device that costs $99. It's going to change every industry.

Nicolas Marchand - Deputy director of GIPSA-lab Directeur de Recherche CNRS - HDR GIPSA-lab, Control Systems Department, Grenoble, France

Biography

Nicolas Marchand received the M.Sc. and Ph.D. degrees in control theory from Grenoble-INP, in 1995 and 1999, Grenoble, France. He is a CNRS researcher and director of GIPSA-lab since 2020, Grenoble, France. His research focuses on event-based control, control and stabilization of flying robots and control theory for computer sciences.

Abstract

Controlling UAV's based on events: a new approach for new solutions

In this talk, we will present the theoretical framework of event-based control. Through examples, we will show how event-based control approach can improve or give new abilities to robotic systems and especially UAVs. Examples will cover safer remote control of UAVs, faster learning for artificial intelligence and other examples related to flying robotics.

Rogelio Lozano - CNRS Research Director Université de Technologie de Compiègne Compiègne, France

Biography

Rogelio Lozano was born in Monterrey Mexico, on July 12, 1954. He received the B.S. degree in electronic engineering from the National Polytechnic Institute of Mexico in 1975, the M.S. degree in electrical engineering from Centro de Investigación y de Estudios Avanzados (CINVESTAV), Mexico in 1977, and the Ph.D. degree in automatic control from Laboratoire d'Automatique de Grenoble, France, in 1981. He joined the Department of Electrical Engineering at CINVESTAV, Mexico, in 1981 where he worked until 1989. He was Head of the Section of Automatic Control from June 1985 to August 1987. He has held visiting positions at the University of Newcastle, Australia, from November 1983 to November 1984, NASA Langley Research Center VA, from August 1987 to August 1988, and Laboratoire d'Automatique de Grenoble, France, from February 1989 to July 1990. Since 1990 he is a CNRS (Centre National de la Recherche Scientifique) Research Director at University of Technology of Compiègne, France. He was Associate Editor of Automatica in the period 1987-2000. He is associate Editor of the Journal of Intelligent and Robotics Systems since 2012 and Associate Editor in the Int. J. of Adaptive Control and Signal Processing since 1988.

He has coordinated or participated in numerous French projects dealing with UAVs. He has recently organized 2 international workshops on UAVs (IFAC RED UAS 2013 and IEEE RAS RED UAS 2015). He participates in the organization of the annual international conference ICUAS (International Conference on Unmanned Aerial Systems) since 2010. He is IPC Chairman of the ICSTCC in Rumania since 2012. He was Head of Heudiasyc Laboratory in the period 1995-2007 and since 2008 He is Head of the Joint Mexican-French UMI 3175 CNRS. His areas of expertise include UAVs, mini-submarines, exo-squelettons and Automatic Control. He has been the advisor or co-advisor of more than 35 PhD theses and published more than 130 international journal papers and 10 books.

Abstract

Stabilization and nonlinear control for a helicopter with virtual swashplate in outdoor flight

The dynamical model of a mini helicopter without swashplate is presented. The helicopter is composed of two rotors, the main rotor with torque modulation and variable pitch propellers to stabilize roll and pitch angles, the second rotor stabilizes yaw displacement. The torque modulation is performed accelerating and decelerating the main rotor which produces a variation in the blades pitch. This helicopter does not have the classical swashplate. The dynamical model is obtained via the Euler–Lagrange approach and a nonlinear control strategy is proposed. The roll and the forward displacement are controlled by using a virtual swashplate. The pitch and lateral displacement are controlled in a similar way. The yaw displacement is stabilized by a classical linear state-feedback controller. The nonlinear controller performance is tested on real experiments using a mini helicopter. It is shown that the controller is robust to disturbances in outdoor flights.

Panels with Former IMAV General Chairs

Panel 1: Former General Chairs of IMAV 2016 and IMAV 2018

Prof. Ben M. Chen
General Chair, IMAV 2016
Beijing, China

Biography

Ben M. Chen is currently a Professor of Mechanical and Automation Engineering at the Chinese University of Hong Kong (CUHK). He was a Provost's Chair Professor in the Department of Electrical and Computer Engineering at the National University of Singapore, before joining CUHK in 2018. He was an Assistant Professor in the Department of Electrical Engineering at the State University of New York at Stony Brook, in 1992–1993. His current research interests are in unmanned systems, robust control and control applications. Dr. Chen is a Fellow of IEEE and Fellow of Academy of Engineering, Singapore. He has authored/co-authored about 500 journal and conference articles, and a dozen research monographs in control theory and applications, unmanned systems and financial market modeling. He had served on the editorial boards of a dozen international journals including Automatica and IEEE Transactions on Automatic Control. He currently serves as an Editor-in-Chief of Unmanned Systems. Dr. Chen has received a number of research awards. His research team has actively participated in international UAV competitions and won many championships in the contests.

Prof. Simon Watkins
General Chair, IMAV 2018
Melbourne, Australia

Biography

Simon Watkins is a Professor of Engineering at RMIT and was Chair of IMAV held at RMIT in 2018. He was listed in the top 2% scientists in the world in 2019 in the field of "Aerospace and Aeronautics", based on Stanford University standardised citation metrics and wrote some of the first papers on atmospheric winds and micro air vehicles. He founded the RMIT Unmanned Aircraft Systems Research Team which comprises a multidisciplinary group of senior academics, research fellows and PhD students. His current research interest is bird and insect flight and trying to reveal the methods by which they maintain steady flight in the turbulent outdoor wind.

Panel 2: Former General Chairs of IMAV 2014, IMAV 2017, IMAV 2019

Prof. Guido de Croon
General Chair, IMAV 2014
Delft, The Netherlands

Biography

Swarms of tiny autonomous drones can help humans in search-and-rescue missions, in keeping track of the stock in warehouses or in monitoring crop in greenhouses. Tiny drones (think below 30 grams) are very safe for humans, are suitable for flying in narrow environments, and are very cheap so that they can be produced in large numbers. However, it is also notoriously hard to make such tiny drones fly autonomously. Due to their extremely low weight, they are very limited in terms of energy and payload. This implies that they can carry few sensors and extremely little processing compared to, e.g., self-driving cars. In my talk, I will talk about the effort at TUDelft's MAVLab to make swarms of tiny autonomous drones, and how we approach this by drawing inspiration from insect intelligence. I will present our work on the lightest autonomous drone in the world, the 20 gram "DelFly Explorer" and on a swarm of 33-gram Crazyflies that is able to autonomously explore an unknown environment and come back to the departure point. Furthermore, I will discuss our recent study in which we designed a swarm of CrazyFlie drones able to autonomously localize gas sources. I will end with our efforts into incorporating spiking neural networks in neuromorphic hardware on our drones, showing results of an evolved spiking neural controller that was successfully ported to the real world, for the first time controlling a flying drone with neuromorphic processing in the control loop.

Prof. Jean-Marc Moschetta
General Chair, IMAV 2017
Toulouse, France

Biography

Jean-Marc Moschetta is a Professor of Aerodynamics at ISAE-SUPAERO, Toulouse, France and Director of the Micro Air Vehicle Research Center. Since 2000, he has devoted his research activity to rotary-wing and fixed-wing drones including : aerodynamic design, energy-harvesting techniques, transitioning vehicles, quiet propellers. Recently, Dr Moschetta has started the development of an Hydrogen-powered fixed-wing UAV for flying over the Atlantic Ocean with low carbon emissions.

Asst. Prof. Gautier Hattenberger
Deputy Chair, IMAV 2017
Toulouse, France

Biography

Gautier Hattenberger is an assistant-professor at the French Civil Aviation University (ENAC) in Toulouse, France. As a member of the UAV Research Program, he is working on flight dynamics and control of micro-UAVs, modeling and simulation, architecture of embedded systems, trajectory planing and formation flight. Most of his work is based on the Open-Source UAV system “Paparazzi”, for which he is now one of the head developer. He graduate from the French national engineering school of aeronautical construction in 2004 and received his Ph.D Degree at the Robotic department of the Laboratory for Analysis and Architecture of Systems (LAAS-CNRS, Toulouse), for his work on formation flight control and planing of UAVs in 2008.

Prof. Pascual Campoy
General Chair, IMAV 2019
Madrid, Spain

Biography

Pascual Campoy is Full Professor on Automatics at the Universidad Politécnica Madrid UPM (Spain) where he lectures on Control, Machine Learning and Computer Vision. He has been visiting professor at DCSC Department in TUDelft (The Netherlands) from 2014 to 2019, and previously visiting professor at Tong Ji University (Shanghai-China) in 2013 and Q.U.T. (Australia) 2011. He received his PhD in Automatics & Robotics at Universidad Politecnica Madrid in 1988, where he previously received his Master tittle in Automatics Engineering in 1983. He is leading the Research Group on “Computer Vision and Aerial Robotics” at U.P.M. within the Centre of Automatics and Robotics (C.A.R.), whose activities are aimed at increasing the autonomy of the Unmanned Aerial Vehicles (UAV) by exploiting the powerful sensor of Vision, using cutting-edge technologies in Image Processing, Control and Artificial Intelligence. He has been heading director of over 40 R&D projects, including R&D European projects, national R&D projects and over 25 technological transfer projects directly contracted with the industry. He is author of around 200 international scientific publications and nine patents, three of them registered internationally. He is awarded in the top international UAV competitions: IMAV12, IMAV13, IARC14, IMAV16 and IMAV17, General Chair for IMAV 2019 and he coordinated the international team that was awarded in the third place in the Grand Challenge MBZIRC20.

Papers

Best Paper Awards and Special issues

For this IMAV 2021, three prizes were awarded during the Awards Ceremony: Best Conference Paper, two runners-up were also mentioned and received a certificate; Best Technical/Application Paper; and Best Student Paper. These 5 papers were selected to be published in a Special Issue of the International Journal of Micro Air Vehicles (Sage).

Best Conference Paper

Nonlinear model predictive control for improving range-based relative localization by maximizing observability.
Shushuai Li, Christophe De Wagter and Guido de Croon. [2] on page 28.

Runners-Up

Estimating wind using a quadrotor.
Gautier Hattenberger, Murat Bronz and Jean-Philippe Condomines. [15] on page 124.

Extended Incremental Non-linear Control Allocation on the TU Delft Quadplane.
Jan Karssies and Christophe De Wagter. [9] on page 74.

Best Technical/Application Paper

Field report: deployment of a fleet of drones for cloud exploration.
Gautier Hattenberger, Titouan Verdu, Nicolas Maury, Pierre Narvor, Fleur Couvreux, Murat Bronz, Simon Lacroix, Grégoire Cayez and Gregory Roberts. [13] on page 109.

Best Student Paper

Design of aeroacoustically stealth MAV rotors. *Pietro Li Volsi, David Gomez-Ariza, Thierry Jardin, Romain Gojon and Jean-Marc Moschetta.* [19] on page 153.

Selection of papers for Unmanned Systems

In addition to the papers above, the following papers were selected to be published in a Special Issue of Unmanned Systems (World Scientific).

Onboard Time-Optimal Control for Tiny Quadcopters
Jelle Westenberger, Christophe De Wagter and Guido C.H.E. de Croon [11] on page 93.

Developing a modular tool to simulate regeneration power potential using orographic wind-hovering UAVs
Midas Gossye, Sunyou Hwang and Bart Remes [14] on page 116.

Framework and evaluation methodology for Autonomous Drone Racing
Miguel Fernandez-Cortizas, Pablo Santamaria, David Perez-Saura, Javier Rodriguez-Vazquez, Martin Molina, Pascual Campoy [5] on page 50.

Position controller for a flapping-wing drone using UWB
*Guillermo González * , Guido C.H.E de Croon, Diana Olejnik and Matěj Karásek* [10] on page 85.

Immersion and Invariance Based Trajectory Tracking Control of an Aerial Manipulation System
Aaron Lopez, Hugo Rodríguez Cortés, Israel Cruz Vega and Jose Martinez-Carranza. [8] on page 68.

Authors

Release

Released on November 19th 2021.

V1.1, Rev.15231, 2021-11-24 09:42
Created by: J. Martinez-Carranza & C. De Wagter

Flight Code Convergence: Fixedwing, Rotorcraft, Hybrid

D.C. van Wijngaarden*, E.J.J.Smeur, and B.W.D. Remes
Delft University of Technology, Kluyverweg 1, Delft, The Netherlands

Abstract

Rotorcraft, fixed wing and hybrid Unmanned Air Vehicles (UAV) each have applications in which they excel. Traditionally, dedicated autopilot control code is written to accommodate flight of each UAV type. This causes fragmentation of control code and may lead to performance differences or errors. In this paper, we propose to use the same INDI controller for rotorcraft, fixed wing and hybrid UAVs, with only parametric differences in control effective matrix definitions and roll, pitch and airspeed limits. The controller is based on earlier work, but relevant derivations are included in this paper. Successful test flights, performed with a Bebop2 quadrotor, a Disco fixed wing, and a Nederdrone tailsitter hybrid demonstrate the feasibility of this approach.

1 Introduction

The amount of applications for Unmanned Air Vehicles (UAVs) has drastically increased over the last couple of years [1]. To best serve their purpose, different applications require different types of drones: fixed wing, rotorcraft, or hybrid. Fixed wing aircraft have superior endurance, while rotorcraft have more flexible maneuvering and hovering capabilities. Hybrid UAVs take the middle ground in terms of endurance and flexible maneuvering.

In the UAVs that are flown, one can broadly make the distinction between commercial UAVs that have proprietary software that is dedicated to one specific UAV type, and open source autopilot systems, that provide flight code for a variety of different UAV types. Examples of the latter are PX4 [2], Ardupilot [3], and Paparazzi [4]. These open source autopilot systems have found a broad user base with universities, amateur drone pilots and startup companies, and each of these autopilot systems supports various fixed wing, rotorcraft and hybrid drone types.

However, the control and guidance code for these different types of UAV, is typically separated. For example, the aforementioned autopilot systems contain dedicated control code for fixed wing, VTOL, and rotorcraft drones. Even the control of hybrid, or transitioning, drones are often done by switching from one controller to another as the vehicle transitions to forward flight [5, 6]. There have been researchers presenting an more integrated, or unified control structure for hybrid UAVs, making use of Nonlinear Dynamic Inversion (NDI) [7] or Incremental Nonlinear Dynamic Inversion (INDI) [8], but this was always aimed at one specific hybrid vehicle.

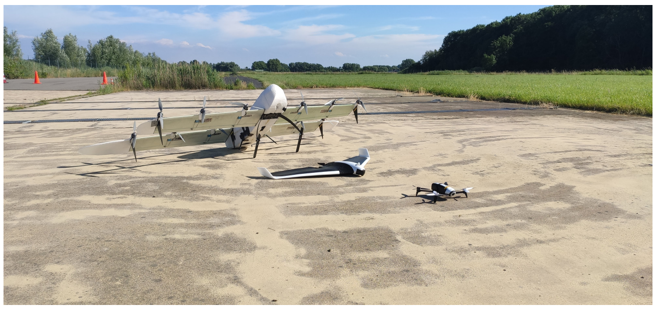


Figure 1: UAV platforms used for the experiments described in this paper. From left to right: Nederdrone (hybrid), Parrot Disco (fixed wing) and a Parrot Bebop2 (rotorcraft).

This fragmentation of code can result in implementation differences and therefore performance differences between the different UAV types. Next to that, keeping all control code up to date requires extra work in code maintenance, which also increases the chance of errors.

In this paper, we demonstrate that it is possible to fly fixed wing, rotorcraft, and hybrid UAVs using the same uniform INDI control and guidance algorithm, with only parametric changes in the control effectiveness matrix definitions and flight envelope protection concerning pitch, roll and airspeed limits. This is achieved through a cascaded INDI controller, based on [9] and [10], that controls attitude and position through closed loop control of angular and linear accelerations. Different vehicle configurations are accommodated through parametric changes of (1) the control effectiveness and (2) flight envelope limits. The flight code used for these experiments is publicly available on Github [1].

In the following sections, a universal controller is outlined that is applied to a Bebop2 quadrotor (rotorcraft), a Disco fixed wing and a Nederdrone (hybrid UAVs). These platforms are described in Section 2. In Section 3 the inner (attitude) controller is explained. Section 4 describes the

*Email address(es): D.C.vanWijngaarden@tudelft.nl

[1] https://github.com/tudelft/paparazzi/tree/convergence

outer loop (position) controller, and highlights the parametric differences required for the different UAV types. Section 5 presents test flights of a rotorcraft, fixed wing and hybrid UAV, using the same control code. Conclusions are drawn in Section 6.

2 Test platforms

Though the controller described in this paper is applicable to a broad range of rotorcraft, fixed wing and hybrid UAVs, we will consider three platforms in particular. These are the Parrot Bebop2 quadrotor, the Parrot Disco fixed wing, and the Nederdrone developed by the MAVLab in Delft [11], all depicted in figure 1. The left UAV in the figure is the Nederdrone, a hybrid UAV that can hover, can transition 90 degrees to fly forward horizontally and take-off and land vertically. The Nederdrone is a biplane tail-sitter with 12 motors and 8 control surfaces distributed over its wings.

The middle UAV is the Parrot Disco, a fixed wing UAV that can only fly forward and does not have hovering capabilities. The parrot Disco is a flying wing that has a single motor and a pair of elevons. These control surfaces can be used for pitch and roll control. The vehicle is passively directionally stable.

Finally, the right UAV is a Parrot Bebop2, a regular quadrotor without any wing surface. The Bebop2 has 4 motors that can provide control inputs for pitch, roll and yaw. The Bebop2 does not have an airspeed sensor, as opposed to the other two vehicles, which use it while flying ”forward”.

For all vehicles, the body reference frame is defined with the Z axis in the opposite direction as the thrust, the Y axis pointing to the right and the X axis completing the right handed axis system (for the fixed wing orthogonal to the wing surface, for the quadrotor through the nose). The body reference frame is illustrated by figure 2.

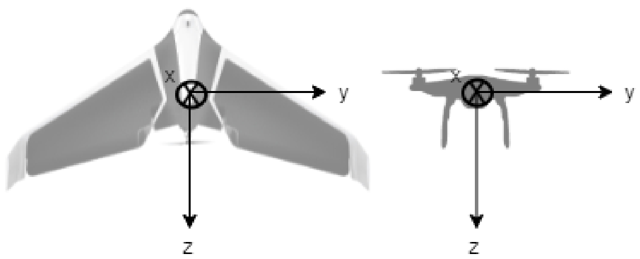


Figure 2: Body reference frame for a fixed wing and a multirotor.

3 INDI inner loop control

The inner loop controller is implemented along the lines of [12], but without the online control effectiveness estimation. For details, we refer to that paper, but for completeness, the controller will be summarized in this section.

The model on which the controller is based is given by the equation for the translational dynamics:

$$\ddot{\boldsymbol{\xi}} = \boldsymbol{g} + m^{-1}\left(\boldsymbol{M}_{NB}\boldsymbol{f} + \boldsymbol{f}_{\text{ext}}\right), \tag{1}$$

where $\boldsymbol{\xi}$ is the position vector in the North East Down (NED) frame, $\boldsymbol{g}$ is the gravity vector in the NED frame and m is the mass of the drone. $\boldsymbol{M}_{NB}$ is the rotation matrix from body to NED frame, which is obtained from the attitude quaternion $\boldsymbol{q}$. $\boldsymbol{f}$ is the input force due to changes in attitude and thrust level and $\boldsymbol{f}_{\text{ext}}$ is an unmodeled external force.

The rotational dynamics are given by:

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \otimes \begin{pmatrix} 0 \\ \boldsymbol{\Omega} \end{pmatrix}, \tag{2}$$

$$\dot{\boldsymbol{\Omega}} = \boldsymbol{J}^{-1}(\boldsymbol{m} + \boldsymbol{m}_{\text{ext}} - \boldsymbol{\Omega} \times \boldsymbol{J}\boldsymbol{\Omega}), \tag{3}$$

where $\boldsymbol{\Omega}$ denotes the angular rates of the vehicle and $\otimes$ the Hamilton product. The inertia matrix of the vehicle is denoted by $\boldsymbol{J}$ and is assumed to be diagonal, $\boldsymbol{m}$ is the moment due to the inputs and $\boldsymbol{m}_{\text{ext}}$ is the moment due to unmodeled external moments.

Neglecting the cross-term in Eq. 3, we can approximate the change in angular acceleration of the vehicle due to a change in the input vector $\boldsymbol{u}$ as:

$$\dot{\boldsymbol{\Omega}} - \dot{\boldsymbol{\Omega}}_0 = \boldsymbol{G}_1(\boldsymbol{u} - \boldsymbol{u}_0) + \boldsymbol{G}_2(\dot{\boldsymbol{u}} - \dot{\boldsymbol{u}}_0), \tag{4}$$

where $\boldsymbol{G}_1$ is the control effectiveness matrix (which incorporates the inertia and may also be a function of velocity), and $\boldsymbol{G}_2$ is a control effectiveness matrix that can account for effectiveness in the derivative of $\boldsymbol{u}$, which can occur due to significant propeller inertia. The current angular acceleration $\dot{\boldsymbol{\Omega}}_0$ can be obtained by differentiation of the gyroscope signal. As differentiation will amplify the high frequency vibrations, a low-pass filter has to be applied to all signals that have the subscript 0, which will now receive the subscript f. As $\boldsymbol{u}$ and $\dot{\boldsymbol{u}}$ are inter-dependent, we approximate the derivative in discrete time using the unit delay operator L as $\dot{\boldsymbol{u}} \approx (\boldsymbol{u}(k) - \boldsymbol{u}(k-1))/T_s = (\boldsymbol{u} - L\boldsymbol{u})/T_s$, where T_s is the sample time. This leads to the prediction of $\dot{\boldsymbol{\Omega}}$ based on a new input command u_c:

$$\dot{\boldsymbol{\Omega}} - \dot{\boldsymbol{\Omega}}_f = (\boldsymbol{G}_1 + \boldsymbol{G}_2)(\boldsymbol{u}_c - \boldsymbol{u}_f) + \boldsymbol{G}_2 L(\boldsymbol{u}_c - \boldsymbol{u}_f). \tag{5}$$

This system can be inverted using the pseudo inverse, denoted by $(.)^+$, or by a more sophisticated control allocation scheme, such as Weighted Least Squares (WLS) control allocation [13]. Since this is now a control law to which the angular acceleration is an input, it is denoted as the virtual control $\boldsymbol{\nu}$.

$$\boldsymbol{u}_c = \boldsymbol{u}_f + (\boldsymbol{G}_1 + \boldsymbol{G}_2)^+(\boldsymbol{\nu} - \dot{\boldsymbol{\Omega}}_f + \boldsymbol{G}_2 z^{-1}(\boldsymbol{u}_c - \boldsymbol{u}_f)) \tag{6}$$

Due to the feedback of the angular acceleration, using the control effectiveness of the actuators, disturbances can be counteracted quickly [12, 9].

Since the INDI controller takes care of most of the non-linearities in the system, the reference angular acceleration $\boldsymbol{\nu}$ can simply be obtained through a PD controller. The attitude is represented as a quaternion, and the vector part of the quaternion error with the desired attitude is used for feedback:

$$\boldsymbol{\nu} = K_D(K_P \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix}_e - \boldsymbol{\Omega}), \tag{7}$$

where $\begin{bmatrix} q_x & q_y & q_z \end{bmatrix}_e^T$ is the vector part of the attitude quaternion error.

The gains K_D and K_P can be tuned or derived analytically [9].

There is a difference between the platforms in terms of their inner loop control. The Disco fixed wing UAV cannot fully control its attitude, as it does not have a rudder, but it still uses the same complete attitude controller. To make sure the reference attitude does not drift away from the real attitude in terms of its heading, the heading of the reference is continuously reset to its current heading in the outer loop. This is only done for the Disco, and not for the other UAVs. This would not be necessary for a fixed wing with yaw control.

Besides this, the differences are purely parametric: each vehicle has its own control effectiveness matrices $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$. For the Disco and the Nederdrone, these matrices are a function of airspeed, as the airflow over the control surfaces greatly affects their effectiveness. For the Bebop2, these control effectiveness matrices are static.

4 INDI OUTER LOOP CONTROL

Though it may be apparent that the inner loop of the three types of UAV can be the same, since they all use their actuators to generate moments through which the attitude can be controlled, this is not obvious for the outer loop. A fixed wing UAV uses its wing to generate lift, and has to manipulate the angle of attack and the bank angle in order to maneuver, and the thrust in order to accelerate in the direction of flight. A rotorcraft on the other hand, can manipulate the amount of lift directly with its propellers, and has to tilt this lift vector in the desired direction of acceleration in order to maneuver.

Yet, hybrid (tailsitter) UAVs have demonstrated that it is possible to combine these different ways of flying in one vehicle, using one unified controller. Using appropriate flight envelope protections, it is reasonable to assume this controller can also be applied to pure rotorcraft or fixed wing UAV.

This is achieved using the guidance algorithm as presented in [10]. For completeness, a slightly shortened derivation will be given here. We will make use of the attitude representation in Euler angles, with the ZXY rotation order ($\boldsymbol{\eta} = \begin{bmatrix} \psi & \phi & \theta \end{bmatrix}$) such that the Euler angle derivatives are well defined at -90 degrees pitch. Then, the linear acceleration is given by:

$$\ddot{\boldsymbol{\xi}} = \boldsymbol{g} + \frac{1}{m}\boldsymbol{L}_N(\boldsymbol{\eta}, V) + \frac{1}{m}\boldsymbol{D}_N(\boldsymbol{\eta}, V) + \frac{1}{m}\boldsymbol{T}_N(\boldsymbol{\eta}, T), \tag{8}$$

where $\boldsymbol{L}_N$, $\boldsymbol{D}_N$, $\boldsymbol{T}_N$ are the lift, drag and thrust in the NED frame (denoted with the subscript N). Using the transformation matrix between the body and NED reference frames:

$$\boldsymbol{M}_{NB} = \begin{bmatrix} c\theta c\psi - s\phi s\theta s\psi & -c\phi s\psi & s\theta c\psi + s\phi c\theta s\psi \\ c\theta s\psi + s\phi s\theta c\psi & c\phi c\psi & s\theta s\psi - s\phi c\theta c\psi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix}, \tag{9}$$

the thrust can be written as:

$$\boldsymbol{T}_N = \boldsymbol{M}_{NB} \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} = \begin{bmatrix} (s\theta c\psi + s\phi c\theta s\psi)T \\ (s\theta s\psi - s\phi c\theta c\psi)T \\ c\phi c\theta T \end{bmatrix}, \tag{10}$$

and the lift as:

$$\boldsymbol{L}_N = \boldsymbol{M}_{NB}^{\theta=-\frac{\pi}{2}} \boldsymbol{L}_B(\theta, V) = \begin{bmatrix} s\phi s\psi L(\theta, V) \\ -s\phi c\psi L(\theta, V) \\ c\phi L(\theta, V) \end{bmatrix} \tag{11}$$

assuming that the flight path angle is small, such that the lift vector is only rotated from the vertical by the bank angle. Note that T and $L(\theta, V)$ will typically be negative, since the body Z axis points down.

Using a first order Taylor expansion, by taking partial derivatives with respect to the controlled input variables ($\boldsymbol{v} = [\phi \quad \theta \quad T]^T$), we can arrive at the incremental model:

$$\ddot{\boldsymbol{\xi}} = \ddot{\boldsymbol{\xi}}_0 + \frac{1}{m}\left(\boldsymbol{G}_T(\boldsymbol{\eta}, T) + \boldsymbol{G}_L(\boldsymbol{\eta}, V)\right)(\boldsymbol{v} - \boldsymbol{v}_0), \tag{12}$$

where we have assumed that the drag changes slowly with respect to the other variables such that its influence on the change in acceleration can be neglected, and where the control effectiveness matrices of thrust and lift are given by:

$$\boldsymbol{G}_T(\boldsymbol{\eta}, T) = \begin{bmatrix} \left(\frac{\partial}{\partial\phi}\frac{1}{m}\boldsymbol{T}_N(\phi, \theta_0, \psi_0, T_0)|_{\phi=\phi_0}\right)^T \\ \left(\frac{\partial}{\partial\theta}\frac{1}{m}\boldsymbol{T}_N(\phi_0, \theta, \psi_0, T_0)|_{\theta=\theta_0}\right)^T \\ \left(\frac{\partial}{\partial T}\frac{1}{m}\boldsymbol{T}_N(\phi_0, \theta_0, \psi_0, T)|_{T=T_0}\right)^T \end{bmatrix}^T \tag{13}$$

and

$$\boldsymbol{G}_L(\boldsymbol{\eta}, V) = \begin{bmatrix} \left(\frac{\partial}{\partial\phi}\frac{1}{m}\boldsymbol{L}_N(\phi, \theta_0, \psi_0, V_0)|_{\phi=\phi_0}\right)^T \\ \left(\frac{\partial}{\partial\theta}\frac{1}{m}\boldsymbol{L}_N(\phi_0, \theta, \psi_0, V_0)|_{\theta=\theta_0}\right)^T \\ (\boldsymbol{0})^T \end{bmatrix}^T. \tag{14}$$

Elaborating these control effectiveness functions gives:

$$\boldsymbol{G}_T(\boldsymbol{\eta}, T) = \begin{bmatrix} c\phi c\theta s\psi T & (c\theta c\psi - s\phi s\theta s\psi)T & s\theta c\psi + s\phi c\theta s\psi \\ -c\phi c\theta c\psi T & (c\theta s\psi + s\phi s\theta c\psi)T & s\theta s\psi - s\phi c\theta c\psi \\ -s\phi c\theta T & -c\phi s\theta T & c\phi c\theta \end{bmatrix} \tag{15}$$

and

$$\boldsymbol{G}_L(\boldsymbol{\eta}, V) = \begin{bmatrix} c\phi s\psi L(\theta, V) & s\phi s\psi \frac{\partial}{\partial\theta}L(\theta, V) & 0 \\ -c\phi c\psi L(\theta, V) & -s\phi c\psi \frac{\partial}{\partial\theta}L(\theta, V) & 0 \\ -s\phi L(\theta, V) & c\phi \frac{\partial}{\partial\theta}L(\theta, V) & 0 \end{bmatrix}. \tag{16}$$

$\ddot{\boldsymbol{\xi}}_0$ is measured by the accelerometer, but this sensor also picks up a vibrations in the airframe, which has to be removed with a low-pass filter. Like before, to keep all signals synchronized with the same phase delay, all terms with subscript zero will be filtered and receive subscript f instead. The equation can then be inverted to obtain:

$$\boldsymbol{v} = \boldsymbol{v}_f + m\left(\boldsymbol{G}_T(\boldsymbol{\eta}, T) + \boldsymbol{G}_L(\boldsymbol{\eta}, V)\right)^{-1}(\ddot{\boldsymbol{\xi}}_{\text{ref}} - \ddot{\boldsymbol{\xi}}_f) \quad (17)$$

where $\ddot{\boldsymbol{\xi}}_{\text{ref}}$ is the reference acceleration to track.

The functions $L(\theta, V)$ and $\frac{\partial}{\partial\theta}L(\theta, V)$ still have to be defined for the vehicles with a wing, for a pure rotorcraft they are zero. Recognizing that the lift only needs to be known to compute the effectiveness of rolling, we assume level flight and a simple relationship with the pitch angle:

$$L(\theta, V) \approx L(\theta) = -9.81\sin(-\theta)m \quad (18)$$

where θ is bounded between $-\pi/2$ and 0.

Similarly, we assume that in forward flight the thrust just compensates the drag, and its effect on accelerations other than in the thrust axis is small, such that for T in Eq. 12 we can write:

$$T(\theta) = -9.81\cos(\theta)m \quad (19)$$

where again θ is bounded between $-\pi/2$ and 0.

Finally, the rate of change of the reference ψ is computed in order to reduce sideslip β, using feed forward and feed back:

$$\dot{\psi}_{\text{ref}} = \frac{g\tan(\phi_t)}{V_l} + K_\beta\beta \quad (20)$$

where $\dot{\psi}_{\text{ref}}$ is the rate of change of the heading reference, g is the gravitational constant and V_l is a limited airspeed, with 10 m/s as a lower limit, to avoid unachievable rotations. This term is not relevant for the quadrotor, as sideslip is not detrimental to its lift generation. Still, it can be kept in as it will also not degrade the flight performance of the quadrotor.

Equation 17 provides a control law for the linear accelerations. The reference can again be obtained from a simple linear PD controller, if the goal is to hover at a waypoint. In order to track a certain trajectory, appropriate reference accelerations can be computed. In this paper, additionally the line tracking velocity vector field [10] was used together with a proportional ground velocity controller.

Though the assumptions in this section are quite crude, and the control effectiveness is probably inaccurate, the controller is still able to track linear accelerations and execute a simple flight plan. One can imagine that with a more accurate control effectiveness, the performance may improve. Important to observe is that the INDI controller provides an abstraction layer: a simple linear controller that outputs an acceleration reference can be used on top of it, that does not need to know about the flight mechanism of the vehicle. The changing control effectiveness matrices will account for the different methods of manipulating the acceleration of the vehicle.

4.1 *Flight envelope limits*

Another aspect in which the three types of UAVs are clearly different is their flight envelope. The rotorcraft should not pitch or roll too much, as that will render the vertical component of the thrust too small to support the weight of the drone. The fixed wing should not fly slower than its minimum airspeed, as that will stall the wing, and the reduced dynamic pressure will render the control surfaces ineffective.

To make sure the different UAVs don't exit their respective flight envelopes, different limits are imposed on the controlled flight states of the outer loop INDI controller. These are considered parametric differences.

Limit	Fixed wing	Hybrid	Rotorcraft
Pitch [deg]	[-115,-75]	[-120,25]	[-35,35]
Roll [deg]	[-45,45]	[-30,30]	[-35,35]
Airspeed [m/s]	>10	-	-

Table 1: Flight envelope limits in the outer loop INDI controller.

5 Flight test results

Flight tests have been performed for three types of UAVs discussed by this paper. All flights have been performed with the same type of INDI control method. Each UAV has flown a route over line segments connecting 4 waypoints. For the rotorcraft and hybrid UAV, some static waypoints have been added to the mission to demonstrate hover capabilities of those platforms. Furthermore, those static waypoints force the hybrid UAV to transition from forward to hover flight and vice versa.

5.1 *Horizontal guidance*

Figure 3 depicts the two dimensional top view of the path flown by the Parrot Disco running the INDI outer loop guidance code. The flight path for this flight is directed in clockwise direction. The target airspeed for this flight was set to 12 m/s to prevent the UAV from stalling. The Parrot Disco starts its turn to the next line segment 50 metres before reaching a waypoint as can be seen in the diagram. This parameter was programmed such that the UAV can join its next line segment. It can be seen that the fixed wing UAV platform manages to follow its target lines during its route.

A top view of the path flown by the Parrot Bebop2 quadrotor is depicted in figure 4. It can be seen that target lines and target positions have been defined for this flight. The target lines are being followed during a route whereas target positions are approached, after which the vehicle hovers at position. The direction of flight over the target lines is in clockwise direction. It can be seen that when the UAV approaches a line segment from a target position that it first flies perpendicular to the line segment after which it joins and follows the line segment.

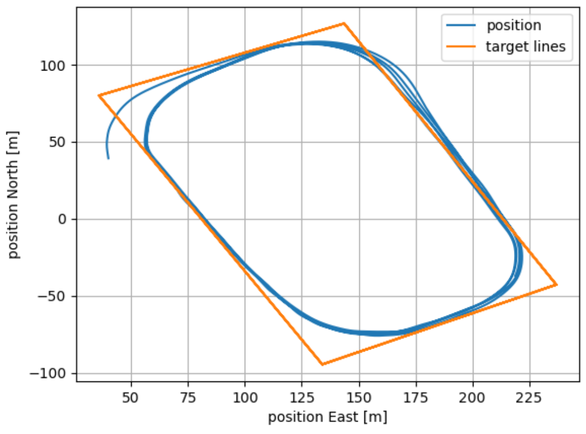


Figure 3: Horizontal position and target lines for the Parrot Disco.

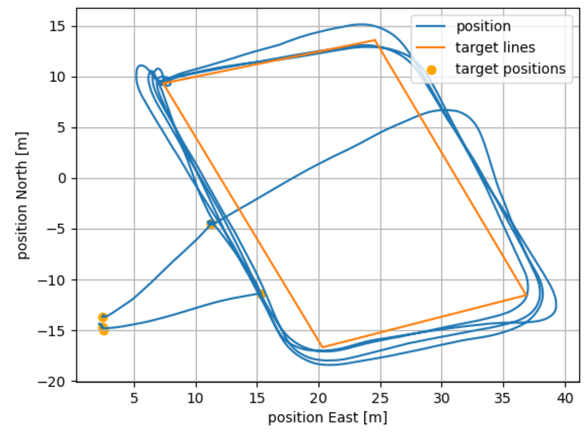


Figure 4: Horizontal position, target lines and target positions for the Parrot Bebop2.

A two dimensional view of the horizontal path flown by the Nederdrone, the hybrid UAV, is given by figure 5. It can can be seen that a mission consisting of target lines and target positions has been flown by this UAV. The flight path connected by the target lines is defined in counterclockwise direction. It can be seen that this platform does not track its lines with the same accuracy as the Disco in forward flight due to the smaller flight envelope in terms of roll angle and a higher airspeed target of 15 m/s in forward flight. However the line tracking accuracy in forward flight is lower for this UAV, the platform has the ability to track target positions with better precision in hover flight as can be seen in the diagram.

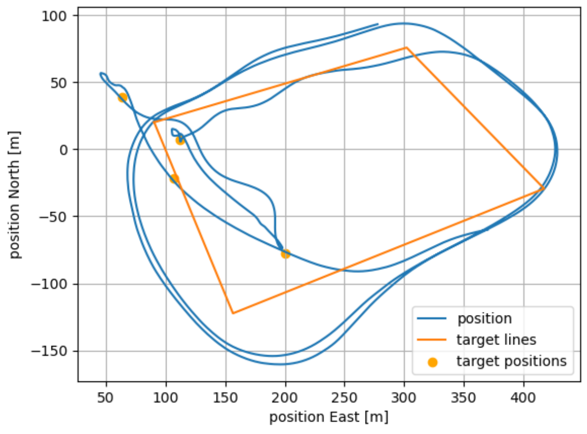


Figure 5: Horizontal position and target lines for the Nederdrone. Includes parts in forward flight, hovering, and complex twisting tailwind transition maneuvers.

5.2 *Vertical guidance*

The test results of the vertical guidance for each type of UAV can be evaluated through an altitude versus time plot as given by figure 6. This plot reflects the altitude and target altitude for each UAV over time.

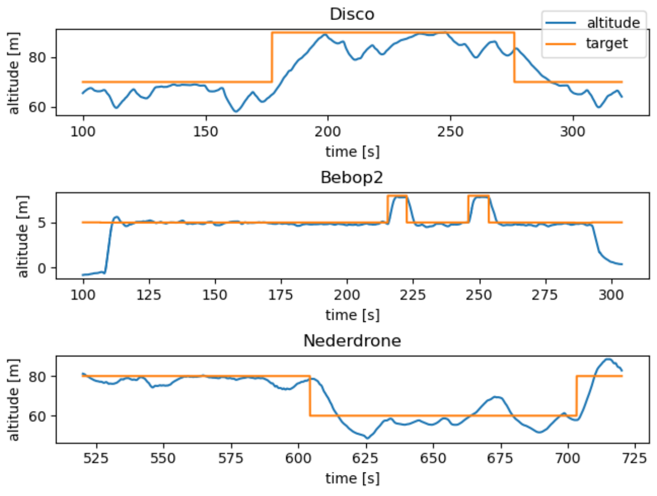


Figure 6: Altitude and target altitude versus time for the Parrot Disco, Parrot Bebop 2 and the Nederdrone.

First of all, it can be seen that the Parrot Disco has a climb and descend in its mission in order to evaluate vertical guidance commanded by the INDI outer loop control. An target altitude is set from 70 metres to 90 metres and back to 70 metres during the route. It is notified that there are some drops in altitude below the target. This occurs during turns which are not compensated for sideslip due to the lack of yaw control for this drone. The drop of altitude is approximately 10 metres below its target.

Secondly, the mission of the Parrot Bebop2 has two climb and descend step inputs between 5 and 7 metres altitude while following a line segment. It can be seen that the tracking of altitude for this platform is more accurate than for a fixed wing due to the direct control of lift by the propellers instead of a lift surface.

Finally, the Nederdrone has a descend and climb step in its mission between 80 and 60 metres altitude. Those climb and descend phases are carried out in forward flight. There is a drop in altitude below its target at 625 seconds. This is at the moment the drone transitions from forward flight to hover flight for which the production of lift is gradually exchanged from the wings to the propellers. The opposite is notified at 675 seconds then the UAV transitions from hover to forward flight.

5.3 *Flight states*

Other flight states that have been logged are the roll angle ϕ and the pitch angle θ using an Euler ZXY rotation order. The airspeed has been logged for the Parrot Disco and the Nederdrone which is used for forward flight.

Diagrams reflecting the pitch and roll angles for all three UAVs are given by figures 7 and 8 respectively. The flight envelope limits per platform as given by table 1 are visualized by dotted lines in those diagrams.

It can be seen that the pitch angle θ stays between the set limits for all platforms. It is notified that the Parrot Disco stays just below its maximum pitch angle limit to prevent the UAV from stalling. The transition of the Nederdrone from forward to hover flight that is initiated at 625 seconds can be seen in the pitch angle plot by an increase in pitch angle. At 710 seconds, the Nederdrone transitions back from hover to forward flight for which the pitch angle is decreased again.

The roll angle plots gives proof that the roll angle ϕ stays within the preset limits for the Parrot Disco and the Parrot Bebop2. The Nederdrone slightly exceeds its bank limit when making turns.

Finally, the airspeed over time for the Parrot Disco and Nederdrone are plotted in figure 9. The target air speeds for forward flight are set to 12 m/s and 15 m/s for the Parrot Disco and Nederdrone respectively.

It can be seen that there are some fluctuations in airspeed for the Disco around its target. This occurs during turns in which sideslip is not compensated due to lack of yaw control for this platform. Therefore the following two effects play a roll that cause fluctuations in airspeed: the airspeed sensor is not aligned with the direction of flight and the drag induced by sideslip affects the airspeed.

It can be seen for the Nederdrone that the target airspeed of 15 m/s is being tracked in forward flight. At the transition around 625 seconds it is notified that the airspeed drops to 0. The hover target position is moved to another place around 675 seconds for which a non zero airspeed is visible on the plot as pitch is being reduced to fly towards the moved location of the target position.

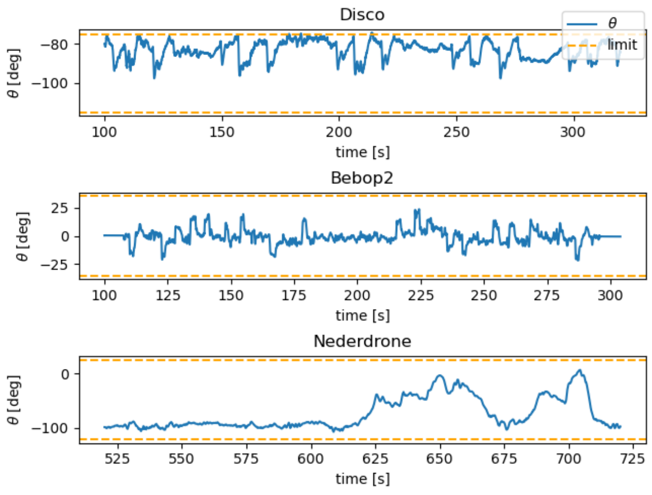


Figure 7: θ (Euler ZXY) versus time for the Parrot Disco, Parrot Bebop2 and the Nederdrone. Dotted lines visualizes the flightenvelope limits.

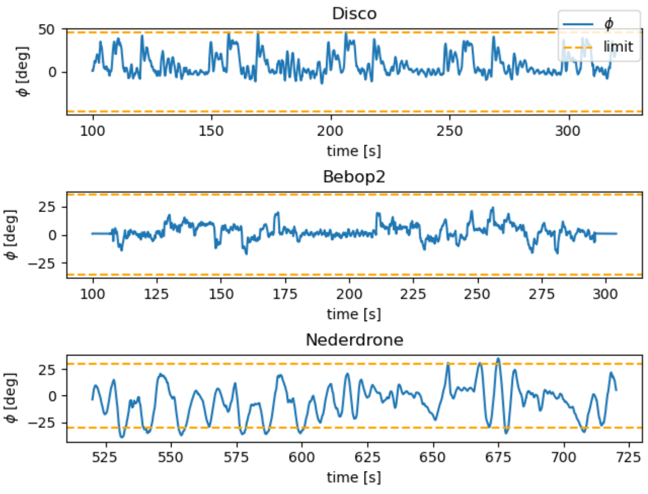


Figure 8: ϕ (Euler ZXY) versus time for the Parrot Disco, Parrot Bebop 2 and the Nederdrone. Dotted lines visualizes the flightenvelope limits.

6 Conclusions

This paper described a cascaded INDI inner and outer loop controller, that is applicable to rotorcraft, fixed wing and hybrid UAVs with only parametric differences. From the successful test flights, we conclude that it is indeed possible to use the same controller for these different UAVs. Parametric flight envelope limits prove to be a simple and effective way of preventing the controller from exiting the flight envelope.

The integrated sideslip controller is appropriate for hybrids and fixed wings, but is not required for multirotors,

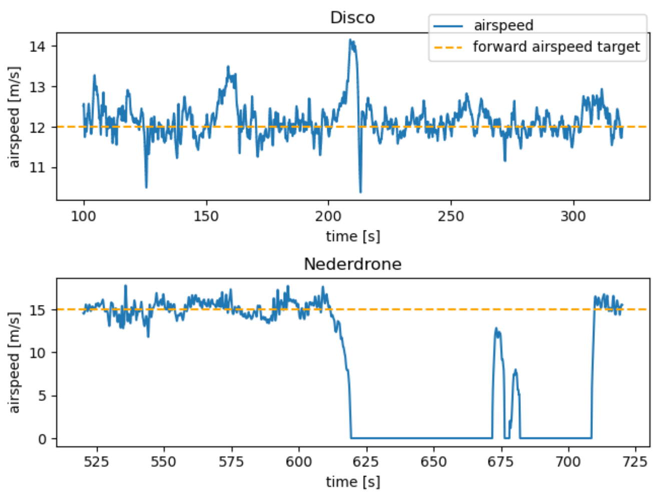


Figure 9: Airspeed versus time for the Parrot Disco and Nederdrone.

which may be constrained in their heading based on the application. This could be accommodated in the future by making this functionality modular. In the future we will show the implementation of this same controller in even more platforms like quad-planes, helicopters and tailed fixed wings. Showing the robustness of this controllers and leading to a single code stack for all types of UAV platforms.

References

[1] Hazim Shakhatreh, Ahmad H. Sawalmeh, Ala Al-Fuqaha, Zuochao Dou, Eyad Almaita, Issa Khalil, Noor Shamsiah Othman, Abdallah Khreishah, and Mohsen Guizani. Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges. *IEEE Access*, 7:48572–48634, 2019.

[2] 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.

[3] Ardupilot. http://www.ardupilot.org/. Accessed: 2021-07-10.

[4] Gautier Hattenberger, Murat Bronz, and Michel Gorraz. Using the Paparazzi UAV System for Scientific Research. In *International Micro Air Vehicle Conference and Competition (IMAV)*, pages 247–252, 2014.

[5] C De Wagter, D Dokter, G de Croon, and B Remes. Multi-Lifting-Device UAV Autonomous Flight at Any Transition Percentage. In *EuroGNC 2013*, pages 1190–1204, 2013.

[6] Boyang Li, Jingxuan Sun, Weifeng Zhou, Chih-Yung Wen, Kin Huat Low, and Chih-Keng Chen. Transition Optimization for a VTOL Tail-sitter UAV. *IEEE/ASME Transactions on Mechatronics*, 2020.

[7] Philipp Hartmann, Carsten Meyer, and Dieter Moormann. Unified velocity control and flight state transition of unmanned tilt-wing aircraft. *Journal of Guidance, Control, and Dynamics*, 40(6):1348–1359, 2017.

[8] Stefan A. Raab, Jiannan Zhang, Pranav Bhardwaj, and Florian Holzapfel. Proposal of a Unified Control Strategy for Vertical Take-off and Landing Transition Aircraft Configurations. In *2018 Applied Aerodynamics Conference*, Reston, Virginia, jun 2018. American Institute of Aeronautics and Astronautics.

[9] Ewoud J. J. Smeur, G.C.H.E. de Croon, and Q. Chu. Cascaded incremental nonlinear dynamic inversion for MAV disturbance rejection. *Control Engineering Practice*, 73:79–90, apr 2018.

[10] Ewoud J. J. Smeur, Murat Bronz, and Guido C. H. E. de Croon. Incremental Control and Guidance of Hybrid Aircraft Applied to a Tailsitter Unmanned Air Vehicle. *Journal of Guidance, Control, and Dynamics*, 43(2):274–287, feb 2020.

[11] C. De Wagter, Bart Remes, Ewoud Smeur, Freek van Tienen, Rick Ruijsink, Kevin van Hecke, and Erik van der Horst. The NederDrone: A hybrid lift, hybrid energy hydrogen UAV. *International Journal of Hydrogen Energy*, pages 1–16, mar 2021.

[12] Ewoud J. J. Smeur, Qiping P Chu, and Guido C H E de Croon. Adaptive Incremental Nonlinear Dynamic Inversion for Attitude Control of Micro Aerial Vehicles. *Journal of Guidance, Control, and Dynamics*, 39(3):450–461, 2016.

[13] Ewoud J.J. Smeur, D ; Höppener, and C. de Wagter. Prioritized Control Allocation for Quadrotors Subject to Saturation. In *IMAV 2017*, pages 37–43, 2017.

Nonlinear model predictive control for improving range-based relative localization by maximizing observability

Shushuai Li[*], Christophe De Wagter and Guido C. H. E. de Croon
Delft University of Technology, Kluyverweg 1, Delft

ABSTRACT

Wireless ranging measurements have been proposed for enabling multiple Micro Air Vehicles (MAVs) to localize with respect to each other. However, the high-dimensional relative states are weakly observable due to the scalar distance measurement. Hence, the MAVs have degraded relative localization and control performance under unobservable conditions as can be deduced by the Lie derivatives. This paper presents a nonlinear model predictive control (NMPC) by maximizing the determinant of the observability matrix in order to generate optimal control inputs, which also satisfy constraints including multi-robot tasks, input limitation, and state bounds. Simulation results validate the localization and control efficacy of the proposed MPC method for range-based multi-MAV systems with weak observability, which has faster convergence time and more accurate localization compared to previously proposed random motions.

1 INTRODUCTION

The use of multiple aerial robots has been studied deeply in recent years for more complicated tasks and challenging environments [1]. For example, a predictive control is proposed for flights of a swarm of five quadrotors despite cluttered obstacles [2]. In outdoor confined spaces, multiple drones are controlled with the evolutionary optimization method for flocking flights [3]. Multiple flying robots coordinate with simultaneous localization based on ranging measurements with beacons [4]. These recent studies show the state-of-art aerial swarm methods. However, most of them rely on extra positioning systems such as indoor optiTrack [2], outdoor GPS [3] or beacons [4].

To remove the dependence of the external infrastructure such as positioning systems, onboard sensors are deployed for developing an autonomous swarm of drones. For example, 3D relative direction can be estimated by sound-based microphone arrays and allows for leader-follower flights of micro aerial vehicles [5]. An array of infrared sensors can also enable relative positioning and inter-robot spatial-coordination [6]. However, these sensor arrays are too heavy and power-consuming for tiny flying robots. In [7], fully distributed and autonomous multiple tiny flying robots explore unknown environments with finite state machine. However, the relative localization is not very accurate due to the direct usage of signal strength, which may not fulfil the precise cooperative tasks.

Vision is the most widely used solution for multi-robot relative localization. Outdoor flocking of multiple drones localize each other with deep neural network and cameras for a safe navigation [8], which requires heavy AI hardware to run the deep network, also for [9] and [10]. Marker-based localization requires simple computation such as recognizing black circles [11] or April tags [12]. But these visual methods are easily influenced by the field of view or lighting conditions that lead to detection failure and localization disaster.

Wireless ranging sensors provide omnidirectional and low-cost ranging measurements, and recently have been used frequently for relative localization. It was initially proposed in [13], where use was still made of Bluetooth in order to fit on tiny MAVs. In [14], an ultra-wide band (UWB) based cooperative relative localization was proposed to estimate the neighbor drones' position based on the distance and self-displacement measurements under common orientation. Furthermore, [15] removes the orientation assumption and achieves the relative localization purely using the distance measurement and acceleration model. However, these experiments assumed high-order dynamic model and has low ranging frequency, which is not efficient for a large number of tiny robots.

In [16], a simplified velocity model and robust ranging protocol are designed for multiple tiny flying robots with self-regulated localization convergence. However, the initialization procedure with random velocity inputs is not efficient. Thus, this paper considers using nonlinear MPC to design the multi-robot controller by maximizing the task performance and degree of observability, while satisfying the constraints such as input velocity bounds and state bounds.

There are some related papers discussing the control of bearing-based or rang-based multi-robot systems [17]. Most papers use persistent excitation methods by setting specific active control patterns to maintain observability, which is not flexible nor optimal for other tasks or constraints.

The main contribution of this paper is leveraging weak

[*]Email address: s.li-6@tudelft.nl

observability theory to optimize the multi-robot control inputs, which has not yet been presented, to the best knowledge of authors. Specifically, the proposed NMPC framework maximizes the nonlinear observability condition derived by Lie derivatives, which is coupled with the velocity inputs and relative states. This leads to faster localization convergence and higher estimation accuracy even after convergence, compared to the random control inputs.

The rest of the paper is organized as follows. Section 2 states the problem including the range-based multi-MAV model, weak observability condition and the problem definition. Section 3 proposes the nonlinear MPC method with the cost function and corresponding constraints. Section 4 gives the simulation results of the proposed control with Acados, an integrated nonlinear MPC tool. The conclusion is discussed in Section 5.

2 Preliminaries

This section briefly introduces the multi-MAV kinematic model and relative Kalman filter. Based on the relative model and distance observations, the observability matrix is determined with Lie derivatives. Finally, the control problem is defined by considering both the model and observability.

2.1 *Relative multi-MAV model*

The model of twin MAVs is described in this subsection, as the relative localization is distributed and triggered by the ranging event among arbitrary two MAVs. The simulated relative model has been tested in real experiments in our previous work, thus it has a small gap compared to the real-world multi-robot system. For details, consult in [16].

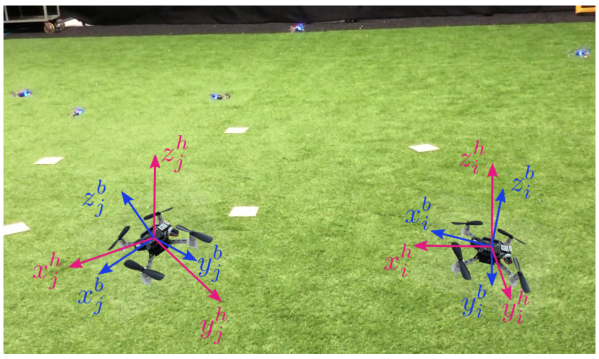


Figure 1: The diagram of a twin-MAV kinematic model, and two coordinated frames. Body frames and horizontal frames are shown with blue axes and red axes, respectively. Both frames are fixed to the robot, while the horizontal frames always have a vertical Z axis. The background images shows previous experiments of multi-MAV relative localization but without optimal control.

For simplicity, we assume the yaw rate of both robots to be zero. This assumption has no influence on the 3D movements of each robot. The control input vector $\boldsymbol{u} = [v_i^x, v_i^y, v_j^x, v_j^y]^T$ represents the XY-axis velocities of the i^{th} and j^{th} robots in their horizontal frames as shown in Fig. 1. The velocities in horizontal frame can be obtained by rotating the measured velocities in body frame, so that the Z axis in the horizontal frame aligns with gravity. The relative state is denoted by $\boldsymbol{x} = [x_{ij}, y_{ij}, \psi_{ij}]^T$, which represents the j^{th} robot's position and relative yaw in the horizontal frame of the i^{th} robot.

The nonlinear relative kinematic model can be derived from Newton formulas by considering the states $\boldsymbol{x}$ and velocity inputs $\boldsymbol{u}$, which can be written as follows [16]

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} cos(\psi_{ij})v_j^x - sin(\psi_{ij})v_j^y - v_i^x \\ sin(\psi_{ij})v_j^x + cos(\psi_{ij})v_j^y - v_i^y \\ 0 \end{bmatrix}. \tag{1}$$

A distance measurement d comes from the DWM1000 ranging sensors, and has the following relation to model states:

$$d = h(\boldsymbol{x}) = \sqrt{x_{ij}^2 + y_{ij}^2 + (h_j - h_i)^2}, \tag{2}$$

where h_i and h_j are the altitudes measured directly from the height sensors. The function $h(\cdot)$ represents the scalar nonlinear observation.

2.2 *Relative estimation*

This subsection briefly reviews the Extended Kalman filter (EKF) for the relative localization. The discrete prediction is formulated as:

$$\begin{aligned} \hat{\boldsymbol{x}}_{k+1|k} &= F(\hat{\boldsymbol{x}}_k, \boldsymbol{u}_k) = \hat{\boldsymbol{x}}_k + \dot{\boldsymbol{x}}_k \Delta t, \\ \boldsymbol{P}_{k+1|k} &= \boldsymbol{A}_k \boldsymbol{P}_{k|k} \boldsymbol{A}_k^T + \boldsymbol{B}_k \boldsymbol{Q}_k \boldsymbol{B}_k^T \end{aligned} \tag{3}$$

where Δt is the update interval, $\boldsymbol{P}$ is the error covariance, $\boldsymbol{A} = \partial F / \partial \boldsymbol{x}$ and $\boldsymbol{B} = \partial F / \partial \boldsymbol{u}$ are the Jacobians of states and inputs, and $\boldsymbol{Q}$ is the process noise covariance.

The final state estimation is estimated by using the distance measurement as shown below:

$$\begin{aligned} \boldsymbol{K}_k &= \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^T (\boldsymbol{H}_k \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^T + \boldsymbol{R}_k)^{-1}, \\ \hat{\boldsymbol{x}}_k &= \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k (d_k - \boldsymbol{H}_k \hat{\boldsymbol{x}}_{k|k-1}), \\ \boldsymbol{P}_k &= (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{P}_{k|k-1} \end{aligned} \tag{4}$$

where $\boldsymbol{K}$ is the Kalman gain, $\boldsymbol{H} = \partial h(\boldsymbol{x}) / \partial \boldsymbol{x}$ is the observation Jacobian, $\boldsymbol{R}$ is the observation noise covariance, and $\boldsymbol{I}$ is the identity matrix.

Remark 1 *The kinematic model and EKF-based relative localization have been validated in real-world experiments [16].*

2.3 *Observability constraint*

Observability of nonlinear systems can be analyzed by Lie derivatives [18]. The corresponding observability matrix is defined as

$$\boldsymbol{O} = [\nabla \mathcal{L}_f^0 h, \nabla \mathcal{L}_f^1 h, \nabla \mathcal{L}_f^2 h]^T \tag{5}$$

where $\mathcal{L}_f h$ means the Lie derivative of function f. The iterations satisfy three conditions: 1) $\mathcal{L}_f^0 h = h(\boldsymbol{x})$; 2) $\mathcal{L}_f^{i+1} h = \nabla \mathcal{L}_f^i h \cdot f$; 3) $\nabla \mathcal{L}_f^i h = \partial \mathcal{L}_f^i h / \partial \boldsymbol{x}$.

Therefore, the relative states are observable only when the observability matrix $\boldsymbol{O}$ is full rank. That means that the determinant should be non-zero, which is expressed as:

$$\begin{aligned}|\boldsymbol{O}| = f_O(\boldsymbol{x}, \boldsymbol{u}) = &-2[-v_i^x v_j^x s(\psi) + v_i^y v_j^x c(\psi) \\ &- v_i^x v_j^y c(\psi) - v_i^y v_j^y s(\psi)] * [-v_j^x y_{ij} c(\psi) + v_j^y y_{ij} s(\psi) \\ &+ v_i^x y_{ij} + v_j^x x_{ij} s(\psi) + v_j^y x_{ij} c(\psi) - v_i^y x_{ij}]\end{aligned} \tag{6}$$

where $s(\cdot)$, $c(\cdot)$, and ψ are simplifications of $sin(\cdot)$, $cos(\cdot)$, and ψ_{ij}, respectively.

2.4 Problem statement

The optimal control problem P_O with respect to observability $|O|$ for this multi-MAV system is defined as:

$$\max_{u^*, k \in \{1,2...\mathrm{N}\}} P_O(|O_k|) = \sum_{k=1}^{\mathrm{N}} |f_O(\boldsymbol{x}_k, \boldsymbol{u}_k)| \tag{7}$$

where $\boldsymbol{u}^*$ is the optimal control vector at current time, which is normally taken from an control sequence. The appropriate control input sequence guarantees the strong observability of the multi-robot system in the future, which can improve the relative localization in both convergence speed and estimation accuracy. The problem is how to calculate the optimal control input sequence.

3 METHODOLOGY

This section proposes a nonlinear model predictive control for solving the optimal problem as described in (7). Then the cost function is further extended for multi-robot tasks such as formation control and motion tracking. In the end, the solver settings for the nonlinear problem (NLP) are presented.

3.1 Nonlinear MPC

The intuitive solution for NLP is MPC, which can achieve the target by minimizing the cost function. Hence, the nonlinear MPC for the proposed problem is designed as follows

$$\boldsymbol{u}_{0|t} := \min_{\boldsymbol{x}_{\cdot|t}, \boldsymbol{u}_{\cdot|t}} J(\boldsymbol{x}_{\cdot|t}, \boldsymbol{u}_{\cdot|t}) \tag{8a}$$

$$\text{s.t. } \boldsymbol{x}_{k+1|t} = f(\boldsymbol{x}_{k|t}, \boldsymbol{u}_{k|t})\delta_t + \boldsymbol{x}_{k|t}, \tag{8b}$$

$$\boldsymbol{x}_{0|t} = \mathrm{sat}_{\boldsymbol{x}_\mathrm{l}}^{\boldsymbol{x}_\mathrm{u}}(\hat{\boldsymbol{x}}_\mathrm{t}), \tag{8c}$$

$$\|\boldsymbol{p}_{\cdot|t}\|_2 - d_{\text{safe}} \geq 0, \tag{8d}$$

$$v_l \leq v_{i,\cdot|t}^x, v_{i,\cdot|t}^y, v_{j,\cdot|t}^x, v_{j,\cdot|t}^y \leq v_u, \tag{8e}$$

$$p_l \leq x_{ij,\cdot|t}, y_{ij,\cdot|t} \leq p_u \tag{8f}$$

where $\boldsymbol{x}_{\cdot|t}$ and $\boldsymbol{u}_{\cdot|t}$ stands for the sequence of states and control inputs in the prediction horizon. The first control value $\boldsymbol{u}_{0|t}$ is taken as the input for the robots. The relative position is denoted by $\boldsymbol{p} = [x_{ij}, y_{ij}]^T$. Saturation function $\mathrm{sat}()$ clips data with lower bound $\boldsymbol{x}_l$ and upper bound $\boldsymbol{x}_u$.

The overall objective function $J(\boldsymbol{x}_{\cdot|t}, \boldsymbol{u}_{\cdot|t})$ is composed by several cost functions, which will be designed later. The remaining equations represent the constraints, which guarantee that the controller satisfies the system dynamic as (8b), the initial state condition related to the current estimated state as (8c), the safe distance for collision avoidance as (8d), the upper and lower bounds of input velocities as (8e), and the relative position bounds as (8f).

Remark 2 *The constraint of initial state is related to the estimated state which is not correct before localization convergence. Hence, the limitation of initial value is necessary to avoid singularity when solving the NLP. A saturation function is employed to limit the the initial value as shown in* (8c). *This is reasonable as many nonlinear robust MPC methods for systems with uncertain states have their stability proof by assuming bounds on the uncertain state.*

3.2 Cost functions

A nonlinear least square (NLS) method is deployed for minimizing the objective function of (8), which is written as:

$$J(\boldsymbol{x}_{\cdot|t}, \boldsymbol{u}_{\cdot|t}) = J_O(\boldsymbol{x}_{\cdot|t}, \boldsymbol{u}_{\cdot|t}) + J_C(\boldsymbol{x}_{\cdot|t}, \boldsymbol{u}_{\cdot|t}) \tag{9}$$

where $J_O(\boldsymbol{x}_{\cdot|t}, \boldsymbol{u}_{\cdot|t})$ and $J_C(\boldsymbol{x}_{\cdot|t}, \boldsymbol{u}_{\cdot|t})$ represent the reformulated observability cost and multi-robot formation coordination cost, respectively.

To maximize the observability with the NLS method, the observability objective (7) is reformulated as the following cost function.

$$J_O(\boldsymbol{x}_{\cdot|t}, \boldsymbol{u}_{\cdot|t}) = \sum_{k=0}^{\mathrm{N}-1} \omega_O \| \frac{a_O}{f_O(\boldsymbol{x}_k, \boldsymbol{u}_k) + \epsilon_O} \| \tag{10}$$

where ω_O, a_O and ϵ_O denote the constant weight, amplitude of cost value, and a small value preventing the singularity.

The coordination cost of $J_C(\boldsymbol{x}_{\cdot|t}, \boldsymbol{u}_{\cdot|t})$ is designed for multi-robot tasks such as motion tracking. Given the reference position sequence $\widetilde{\boldsymbol{p}}_{\cdot|t} = [\widetilde{x}_{ij,\cdot|t}, \widetilde{y}_{ij,\cdot|t}]^T$, the motion tracking cost function is designed as:

$$J_C(\boldsymbol{x}_{\cdot|t}) = \sum_{k=0}^{\mathrm{N}-1} \omega_C \|\boldsymbol{p}_{k|t} - \widetilde{\boldsymbol{p}}_{k|t}\| \tag{11}$$

where $\boldsymbol{p}_{\cdot|t}$ is the predicted relative position sequence in the proposed MPC, calculated by (8b). Specially, if the reference sequence is constant such that $\widetilde{\boldsymbol{p}}_{\cdot|t} \equiv [a_x, a_y]$, the multi-robot motion tracking reduces to formation control.

Remark 3 *Since the coordination task is inaccurate before the localization convergence, the weight ω_C can be set dynamically for the control stability according to the localization accuracy, e.g., the trace of the estimation error covariance* $\mathrm{tr}(\boldsymbol{P})$. *However, a constant $\omega_C = 2$ is enough for the following formation task.*

Sometimes, a penalty cost can be introduced to smooth the control inputs as follows:

$$J_U(\boldsymbol{u}_{\cdot|t}) = \sum_{k=1}^{\mathrm{N}-1} \omega_U \|\boldsymbol{u}_{k|t} - \boldsymbol{u}_{k-1|t}\| \tag{12}$$

Other multi-robot motion control can also be incorporated into the cost function of $J(\boldsymbol{x}_{\cdot|t}, \boldsymbol{u}_{\cdot|t})$. This paper does not discuss the details of those cost functions such as flocking, swarming, and cooperative coordination.

3.3 Acados solver

The nonlinear MPC solver we use in this paper is Acados, which is an open-source and high-performance library for fast optimal control [19]. This software supports Python and is finely tuned for multiple CPU. As for the model definition and differentiation, CasADi is employed to deal with the constraints and model calculations [20].

The brief process of the solver setting is summarized as below. First, the continuous optimal problem is discretized by the multiple shooting method. Furthermore, real-time iteration (RTI) is selected to solve the sequential quadratic programming (SQP). The corresponding Hessian approximation is based on Gauss-Newton. The quadratic problems (QP) in SQP are solved with the partial condensing HPIPM, which is based on linear algebra library BLASFEO. Overall, this solver has a competitive computation speed compared to other stat-of-the-art NMPC solvers.

4 SIMULATION RESULTS

This section shows the improvement of the proposed nonliear MPC on the relative localization performance compared to the stochastic initialization procedure studied in [16]. The statistics of the localization errors and convergence speed are analyzed to validate the efficiency of the proposed controller. In addition, adaptive formation flight of multiple MAVs is studied as example application.

4.1 Simulation set-up

The following simulation experiments are conducted on a Dell Latitude 7480 laptop with a i7-6600U CPU with 4 cores at 2.60GHz and 8GB of RAM. For the simulation experiments, the corresponding EKF parameters are chosen as $\Delta t = 0.01s$, $t_{\mathrm{sim}} = 40s$, $\boldsymbol{Q} = \mathrm{diag}([0.25, 0.25, 0.01])$, and $\boldsymbol{R} = \mathrm{diag}([0.1])$. The initial estimated relative states are set to zero. In contrast, the initial ground-truth positions of each robot are randomly generated, such that the EKF estimation has no prior knowledge of the initial state information. The error covariance is initialized as $\boldsymbol{P} = \mathrm{diag}([10, 10, 0.1])$.

As for the parameters of the proposed nonlinear MPC, the horizon is set to $\mathrm{N} = 50$ and prediction time to $T_f = 1s$, which means each control prediction takes $\delta_t = 0.02s$. Larger prediction horizon has long-term constraint guarantees but with more computation burden. In the observability

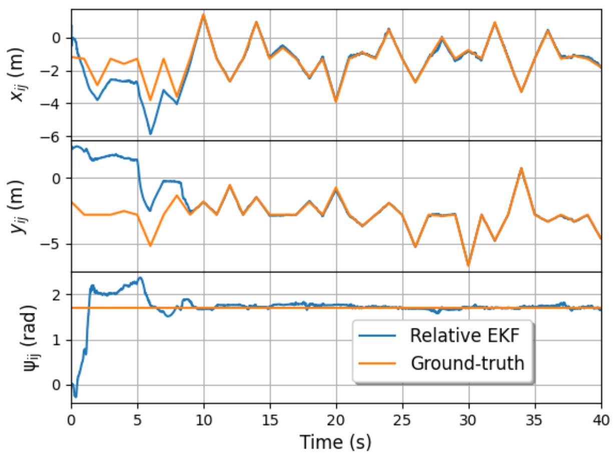


Figure 2: Relative state from EKF estimation and ground-truth between two MAVs under the random velocity inputs. The data consists of 2-axis relative positions and 1-axis relative orientation.

cost function, the parameters are $a_O = 0.021$, $\epsilon_O = 0.001$, and $\omega_O = 1$.

For the constraint settings, the saturation parameters for the initial state vector are chosen as $\boldsymbol{x}_l = [-4, -4, -15]$ and $\boldsymbol{x}_u = [4, 4, 15]$. The safe distance is set to $d_{\mathrm{safe}} = 0.1m$. The velocity input is bounded between $v_l = -2m/s$ and $v_u = 2m/s$. The minimum and maximum relative positions are set to $p_l = -4m$ and $p_u = 4m$, which prevents them flying far from each other.

4.2 Improvement on relative localization

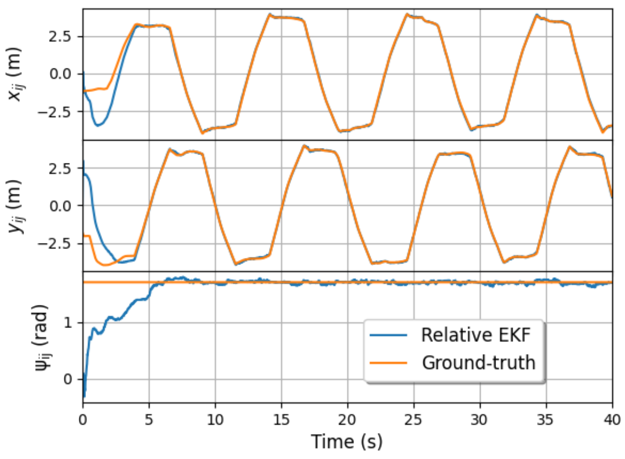


Figure 3: Relative state from EKF estimation and ground-truth between two MAVs under the nonlinear MPC controller. The data consists of 2-axis relative positions and 1-axis relative orientation.

This subsection compares stochastic initialization with nonlinear MPC, in order to verify that the proposed controller with consideration of pure observability cost has better localization performance than the former one. In this subsection, the multi-robot task cost J_C is set to zero.

Fig. 2 and Fig. 3 shows the relative localization performance with the same initial relative states and same parameters for the EKF. Be notified that the three initial states are completely unknown for both the EKF and the controllers. Additionally, the maximum velocities for both controllers are set to be 2 m/s. From these two figures, we can see that the relative positioning with optimal controller has a faster convergence time (about 5s) compared to that of the random controller (about 9s). Especially, observability optimized NMPC has finite-time convergence in the axis of relative yaw, while the random control leads to overshooting as shown in the third subplot of Fig. 2. Therefore, the proposed controller with observability consideration excites all relative states which become more observable even with the unknown initial state errors.

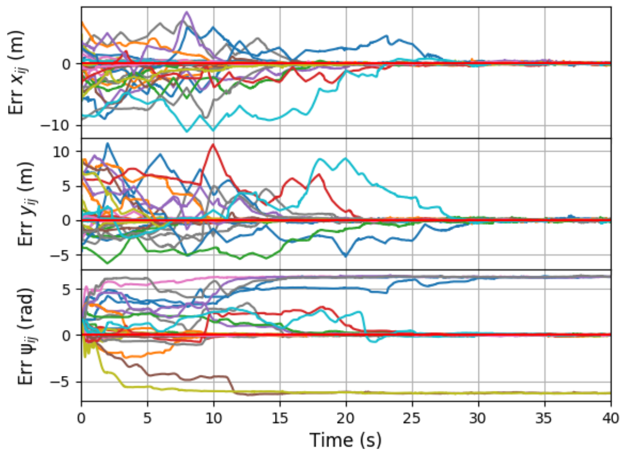


Figure 4: 30 simulation experiments of the stochastic controller from [16] with random initial MAV positions. This figure shows the estimation errors of 2-axis relative positions and 1-axis relative orientation. Note that yaw error with -2π or 2π offset has no influence on the relative localization due to the cos and sin operation.

In addition, after the localization is converged in Fig. 3, the optimal controller automatically generates a periodic motion pattern which is similar to the manual-designed persistent excitation motions. In addition, even with incorrect relative states, they still can avoid each other as shown in Fig. 3, because the observability cost penalizes the collision situation during which the observability determinant approximates zero.

To validate the general efficacy of the proposed NMPC, we gather more statistics on the performance. As shown in Fig. 4 and Fig. 5, 30 random simulation experiments are conducted for each controller. During each simulation epoch, the initial positions for both robots are generated randomly. Moreover, the velocity and distance measurement noise are also created randomly. Both figures imply that the proposed NMPC controller has in general a faster localization convergence speed.

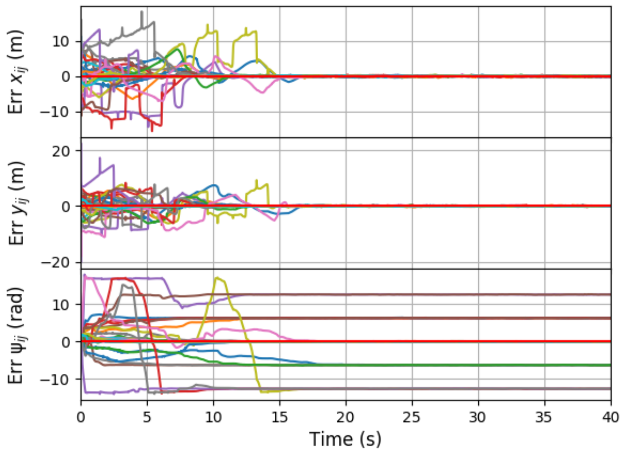


Figure 5: 30 simulation experiments of the optimal controller with random initial MAV positions. This figure shows the estimation errors of 2-axis relative positions and 1-axis relative orientation.

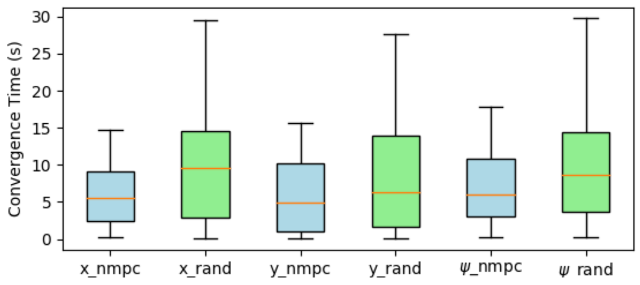


Figure 6: The statistics of convergence time of three-dimensional relative localization under 30 random tests. Blue: the proposed nonlinear MPC; Green: the stochastic control.

Fig. 6 shows the comparison of the detailed convergence time of two controllers with 30 random tests. From it we can see that the average convergence time of the NMPC on all axes is smaller than that of the random controller. Besides, NMPC with observability constraint has a lower maximum convergence time compared to random control inputs.

Another interesting result is the localization accuracy after estimation convergence. Fig. 7 shows the distributions of position estimation errors in the last 5 seconds of two controllers in the 30 random tests. Obviously, the proposed NMPC has lower averaged position estimation errors compared to the stochastic controller. Therefore, the behaviours after convergence are still meaningful to the localization performance. To study it, the control input $\boldsymbol{u}$ for two MAVs is shown in Fig. 8. From which we can see that all 4-channel velocities are approximating the maximum value of 2m/s. The

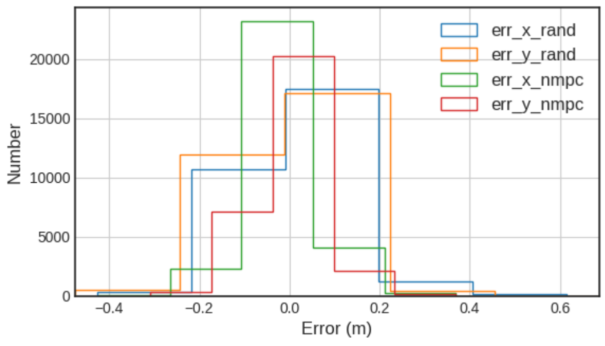


Figure 7: Localization error of two controllers after estimation convergence. Each distribution has total 15000 data on these 30 random tests, which is taken from the last 5 seconds when all estimators have converged.

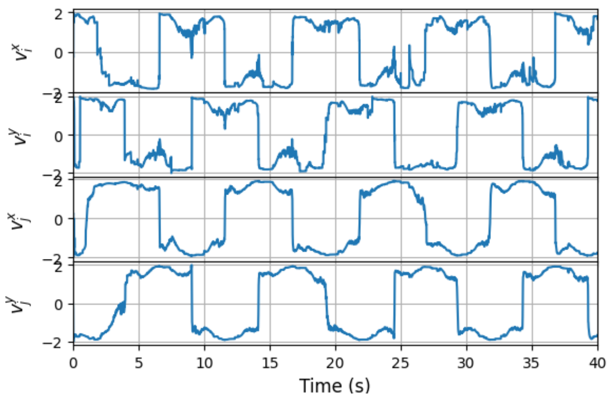


Figure 8: The control inputs generated by the proposed NMPC with observability optimization. These sequences show the velocity input values corresponding to the simulation in Fig. 3.

oscillations and changes of velocity direction occur due to the state bounds and velocity limitation. The four velocities are assigned different phases equally of the periodic motion pattern. This asynchronous behaviour has not been considered before, but NMPC can generate it automatically.

4.3 Formation control with NMPC

This subsection uses the NMPC controller for multi-robot tasks. Examples of formation flight and dynamic motion tracking are given below. At the beginning, a constant relative position is set in the task cost J_C, where $\widetilde{\boldsymbol{p}}_{\cdot|t} = [1, 1]m$. The other settings of the solver remain unchanged. After 15s, a variant relative motion reference is introduced, which is defined as $\widetilde{\boldsymbol{p}}_{\cdot|t} = [2cos(t), 2sin(t)]m$. This leads to a circle motion of the second MAV around the first MAV.

The corresponding control results are shown in the following figures. In Fig. 9 we can see that the proposed NMPC has fast and stable tracking performance given the formation and dynamic tracking tasks at $t = 5$ and $t = 15s$ respectively. In addition, the observability cost keeps being optimized simultaneously by the NMPC.

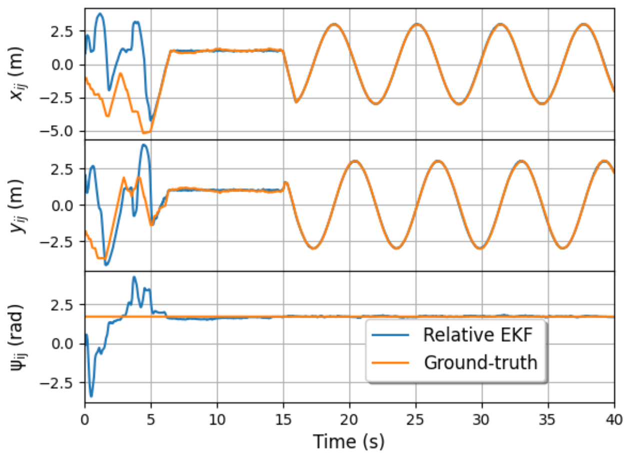


Figure 9: Relative localization and ground-truth between two MAVs under the proposed optimal control method and formation tracking multi-robot tasks. The target relative position is constant before $t = 15s$, and variant after $t = 15s$.

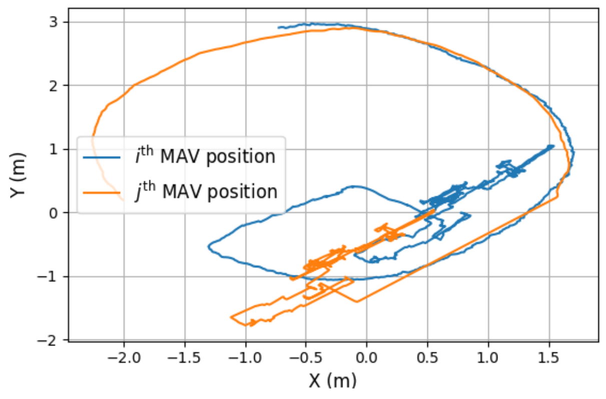


Figure 10: The world-frame trajectories of both MAVs under the proposed optimal control method and formation tracking multi-robot tasks. The time range of the data is between 10s and 20s.

To view the motion of each MAV in world-frame, the trajectories of both MAVs are plotted in Fig. 10. For the formation flight during 10-15s, both MAVs move slowly with constant relative positions. During 15-20s, both MAVs move to achieve the circle tracking and keep optimizing the observability according to the asynchronous behaviours. In addition, from trajetories after 15s in Fig. 9 we can see that introducing the multi-robot task cost eliminates the transients as shown in Fig. 8.

5 Conclusions

This paper proposes a novel nonlinear MPC controller with an observability cost to improve range-based multi-MAV relative localization. Simulation results demonstrate its faster localization convergence and lower estimation errors with respect to previously studied stochastic motion. Future work involves the implementation of this controller in real-world micro air vehicles for better localization and control.

References

[1] M. Coppola, K. N. McGuire, C. De Wagter, and G. C. de Croon, "A survey on swarming with micro air vehicles: Fundamental challenges and constraints," *Frontiers in Robotics and AI*, vol. 7, p. 18, 2020.

[2] E. Soria, F. Schiano, and D. Floreano, "Predictive control of aerial swarms in cluttered environments," *Nature Machine Intelligence*, vol. 3, no. 6, pp. 545–554, 2021.

[3] G. Vásárhelyi, C. Virágh, G. Somorjai, T. Nepusz, A. E. Eiben, and T. Vicsek, "Optimized flocking of autonomous drones in confined environments," *Science Robotics*, vol. 3, no. 20, 2018.

[4] M. Hamer and R. D'Andrea, "Self-calibrating ultra-wideband network supporting multi-robot localization," *IEEE Access*, vol. 6, pp. 22 292–22 304, 2018.

[5] M. Basiri, F. Schill, P. Lima, and D. Floreano, "On-board relative bearing estimation for teams of drones using sound," *IEEE Robotics and Automation letters*, vol. 1, no. 2, pp. 820–827, 2016.

[6] J. F. Roberts, T. Stirling, J.-C. Zufferey, and D. Floreano, "3-d relative positioning sensor for indoor flying robots," *Autonomous Robots*, vol. 33, no. 1, pp. 5–20, 2012.

[7] K. McGuire, C. De Wagter, K. Tuyls, H. Kappen, and G. C. de Croon, "Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment," *Science Robotics*, vol. 4, no. 35, 2019.

[8] F. Schilling, F. Schiano, and D. Floreano, "Vision-based drone flocking in outdoor environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2954–2961, 2021.

[9] H. Xu, L. Wang, Y. Zhang, K. Qiu, and S. Shen, "Decentralized visual-inertial-uwb fusion for relative state estimation of aerial swarm," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 8776–8782.

[10] A. Carrio, S. Vemprala, A. Ripoll, S. Saripalli, and P. Campoy, "Drone detection using depth maps," in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2018, pp. 1034–1037.

[11] J. Faigl, T. Krajník, J. Chudoba, L. Přeučil, and M. Saska, "Low-cost embedded system for relative localization in robotic swarms," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 993–998.

[12] M. Gassner, T. Cieslewski, and D. Scaramuzza, "Dynamic collaboration without communication: Vision-based cable-suspended load transport with two quadrotors," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5196–5202.

[13] M. Coppola, K. N. McGuire, K. Y. Scheper, and G. C. de Croon, "On-board communication-based relative localization for collision avoidance in micro air vehicle teams," *Autonomous robots*, vol. 42, no. 8, pp. 1787–1805, 2018.

[14] K. Guo, Z. Qiu, W. Meng, L. Xie, and R. Teo, "Ultra-wideband based cooperative relative localization algorithm and experiments for multiple unmanned aerial vehicles in gps denied environments," *International Journal of Micro Air Vehicles*, vol. 9, no. 3, pp. 169–186, 2017.

[15] S. van der Helm, M. Coppola, K. N. McGuire, and G. C. de Croon, "On-board range-based relative localization for micro air vehicles in indoor leader–follower flight," *Autonomous Robots*, vol. 44, no. 3, pp. 415–441, 2020.

[16] S. Li, M. Coppola, C. De Wagter, and G. C. de Croon, "An autonomous swarm of micro flying robots with range-based relative localization," *arXiv preprint arXiv:2003.05853*, 2020.

[17] Z. Han, K. Guo, L. Xie, and Z. Lin, "Integrated relative localization and leader–follower formation control," *IEEE Transactions on Automatic Control*, vol. 64, no. 1, pp. 20–34, 2018.

[18] R. Hermann and A. Krener, "Nonlinear controllability and observability," *IEEE Transactions on automatic control*, vol. 22, no. 5, pp. 728–740, 1977.

[19] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "Acados: A modular open-source framework for fast embedded optimal control," *arXiv preprint arXiv:1910.13753*, 2019.

[20] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

Modeling and Identification of Multirotor Drone Dynamics for Onboard MPC Motion Planning

Mathias Bos†*, Bart Theys‡, Jan Swevers† and Goele Pipeleers†
† MECO Research Team, Department of Mechanical Engineering, KU Leuven,
DMMS lab, Flanders Make, Leuven, Belgium
‡ Robotics Research Group, Department of Mechanical Engineering, KU Leuven,
ROB lab, Flanders Make, Leuven, Belgium

ABSTRACT

This paper presents a nonlinear model of multirotor drone dynamics selected specifically to be suited for onboard motion planning using Model Predictive Control (MPC), and presents a generally applicable procedure to identify the model parameters solely based on outdoor flight data. The model stems from a trade-off between prediction accuracy and computational complexity, approximating attitude and thrust control loops as low order linear time-invariant subsystems but without linearizing the thrust vector orientation, and including linear aerodynamic drag and a simplified battery voltage dependency. An open loop simulation compared to recorded data from a free flight maneuver motivates the proposed model complexity in contrast to further simplifications of the proposed model.

1 INTRODUCTION

Despite the current state of computing hardware with a form factor that is convenient for compact drones, onboard real-time motion planning and control remains a challenge. Fully autonomous operation requires state estimation, motion planning and control all to be executed onboard at a reasonable update rate. One of the promising techniques for advanced motion planning and control is Model Predictive Control (MPC). Literature shows this technique already being applied for drone control and navigation, most often restricted to position reference tracking over short horizons using linearized dynamics to limit the computational load [1].
The modeling for simultaneous motion planning and control using MPC asks for a trade-off between sufficient model prediction accuracy and limited model complexity, to reduce the computational load of solving a finite-horizon optimal control problem. In literature, quadrotor models with varying complexity and accuracy have been established, a short review of which now follows.

*Email address: mathias.bos@kuleuven.be
ORCID: 0000-0002-5471-6691

1.1 Related work

A detailed survey of existing kinematic and dynamic models of quadrotors and their derivation is presented in [2]. This survey covers how in general these models can be derived using the Newton-Euler method or the Euler-Lagrange formalism, and how given some assumptions on structure rigidity, symmetry and center of gravity, the most commonly used basic quadrotor model is derived starting from Newton's second law. This basic nonlinear model can be formulated as

$$\begin{cases} \dot{\boldsymbol{p}} &= \boldsymbol{v} \\ \dot{\boldsymbol{v}} &= \boldsymbol{g} + \frac{1}{m}\mathbf{R}\boldsymbol{f}_t \\ \mathbf{J}\dot{\boldsymbol{\omega}} &= \boldsymbol{\tau} - \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}, \end{cases} \tag{1}$$

where $\boldsymbol{p}$ is the drone position vector, $\boldsymbol{v}$ the drone velocity vector, $\boldsymbol{g}$ the gravitational acceleration vector, m the drone mass, $\mathbf{R}$ the rotation matrix from the body to the world frame, $\boldsymbol{f}_t$ the total thrust force, $\boldsymbol{\tau}$ the body torques, $\boldsymbol{\omega}$ the angular rate vector and $\mathbf{J}$ the drone inertia matrix. The thrust force equals the vector sum of all motor thrusts. Often they are assumed to be aligned with the vertical body axis, such that the thrust magnitude equals the sum of the individual motor thrusts. Body torques can be expressed as a function of the thrust forces and the quadrotor geometry.

The set of equations in Equation 1 still misses a relation between the orientation, here represented by the rotation matrix $\mathbf{R}$, and the angular rate vector $\boldsymbol{\omega} = [p, q, r]^\top$. This relation is usually constituted in one of three ways, depending on the used representation of the drone body orientation: through roll-pitch-yaw Euler angles and their derivatives, through quaternion derivatives, or through the derivative of the rotation matrix itself. In terms of the roll-pitch-yaw Euler angles ϕ, θ and ψ, $\mathbf{R}$ is, introducing $c_\gamma \triangleq cos(\gamma)$, $s_\gamma \triangleq sin(\gamma)$, $t_\gamma \triangleq tan(\gamma)$:

$$\begin{aligned} \mathbf{R} &= \mathbf{R}_\psi \mathbf{R}_\theta \mathbf{R}_\phi \\ &= \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi & c_\phi \end{bmatrix}. \end{aligned} \tag{2}$$

The Euler angle derivatives are in that case related to the angular rates by

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi / c_\theta & c_\phi / c_\theta \end{bmatrix} \boldsymbol{\omega}. \tag{3}$$

When using the quaternion rotation formulation $\mathbf{R}(\boldsymbol{q})$, the derivative of the orientation uses the Hamilton quaternion multiplication to express the derivative of the quaternion as $\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \odot \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix}$ [3]. In the third alternative, the derivative of $\mathbf{R}$ can be formulated directly as $\dot{\mathbf{R}} = \mathbf{R}\lfloor\boldsymbol{\omega}\times\rfloor$, where $\lfloor\boldsymbol{\omega}\times\rfloor$ is the skew-symmetric matrix formed with the elements of $\boldsymbol{\omega}$ [4].

The basic model in Equation 1 does not account for aerodynamic effects, which is often sufficient for applications close to the hovering regime. In [2] and [5], a number of aerodynamic effects and ways to model them are listed, such as the ground and ceiling effect, the effect of the angle of attack with respect to the free stream, blade flapping, and interference caused by the vehicle body in the slip stream of the rotor. Often, an approximation that is linear in the translational velocity is used such as in [6], as this describes the dominant aerodynamic effects fairly well even up to significant velocities. The velocity induced aerodynamic drag force is then given by $\boldsymbol{f}_v = -\mathbf{R}\mathbf{D}\mathbf{R}^\top\boldsymbol{v}$, with $\mathbf{D}$ a diagonal matrix containing drag coefficients. This drag force is added to the second line in Equation 1.

A very common approach to avoid the complexity and nonlinearity of the standard nonlinear model, is to perform a linearization around the hover state and assume small angular deviations from this state. Also for drone MPC this is a popular approach, as it alleviates the computational burden, but doing so sacrifices prediction accuracy for more dynamic behavior that deviates significantly from the hover state [1].

Another approach to approximate the full drone model, is to assume that an attitude controller is already in place, which is either assumed to track attitude references perfectly as in [7], or responds as a torsional inertia-spring-damper SISO system in a fully linear rotational and translational model as in [8].

Lastly it can be noted that more complex and detailed models exist, such as [9], in which brushless DC motors and electronic speed controller models are included. However, to the best of our knowledge these detailed models are not used in (online) motion planning and control.

A range of system identification methods is discussed in [2], covering identification procedures for first principles-based models, also referred to as white box models, for (linearized) grey box models, and data-based black box models. Often, however, model parameters are retrieved in a cumbersome procedure from CAD models, which requires detailed models for all parts of the quadrotor, and test benches for separate components such as the motors and propellers [10].

1.2 Contribution and paper structure

This paper proposes a nonlinear multirotor drone model without a small angle assumption that is specifically selected to address the trade-off between model complexity and computational efficiency, together with a simple procedure to identify the unknown model parameters, that does not require special facilities other than the sensors that are present in a realistic outdoor drone application and that do not require dismantling the drone to identify components separately.

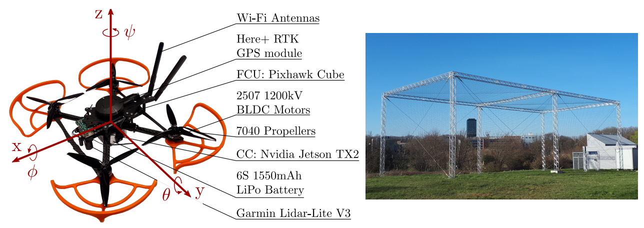


Figure 1: Quadrotor used in the experiments, displaying the axis convention on the body frame with the roll-pitch-yaw Euler angles, and outdoor infrastructure with safety nets.

The paper is structured as follows. First Section 2 presents the model, next Section 3 describes the identification procedure and parameter fitting results, and finally in Section 4 a free flight experiment qualitatively validates the proposed model and identification procedure. This validation compares the simulation of the proposed model to data recorded in the experiment, and compares the simulation results with more simplified versions of the model, justifying the inclusion of nonlinearity, drag and battery voltage dependency. Lastly the limitations to the applicability of the model are discussed.

2 Proposed dynamic model

The proposed model simplifies the ensemble of the drone dynamics and the control cascade of attitude control, angular rate control and motor control into a structure that takes throttle input, roll angle, pitch angle and yaw rate references as input, which is the control level often referred to as the stabilize or angle flight mode in commonly used flight controllers. The choice for attitude-throttle inputs in the proposed model is motivated by the applicability on the popular hardware setup where a Companion Computer (CC) performs high-level optimal control and sends reference inputs to the Flight Control Unit (FCU)[1]. This setup shows a significant communication and processing delay in the order of 0.1 s and the update frequency of state estimates from the FCU to the CC is limited at $50-100$ Hz, rendering it infeasible to perform attitude control which requires a control rate in the order of 400 Hz [11]. Control with attitude reference inputs however has been demonstrated with input frequencies around 50 Hz in [8] and [12], which is realistic for the given setup and still allows dynamic control.

The proposed model is schematically shown in Figure 2. The schematic reads from the top, starting with the control inputs, to the bottom, towards the output which is the drone

[1]Computation hardware used for this paper:
FCU: Pixhawk 2.1 Cube Black running ArduCopter 3.7-dev firmware. (`https://github.com/unl-nimbus-lab/ardupilot`)
CC: Nvidia Jetson TX2 running Jetpack 4.3.
Serial communication with MAVLink using the Mavros library.

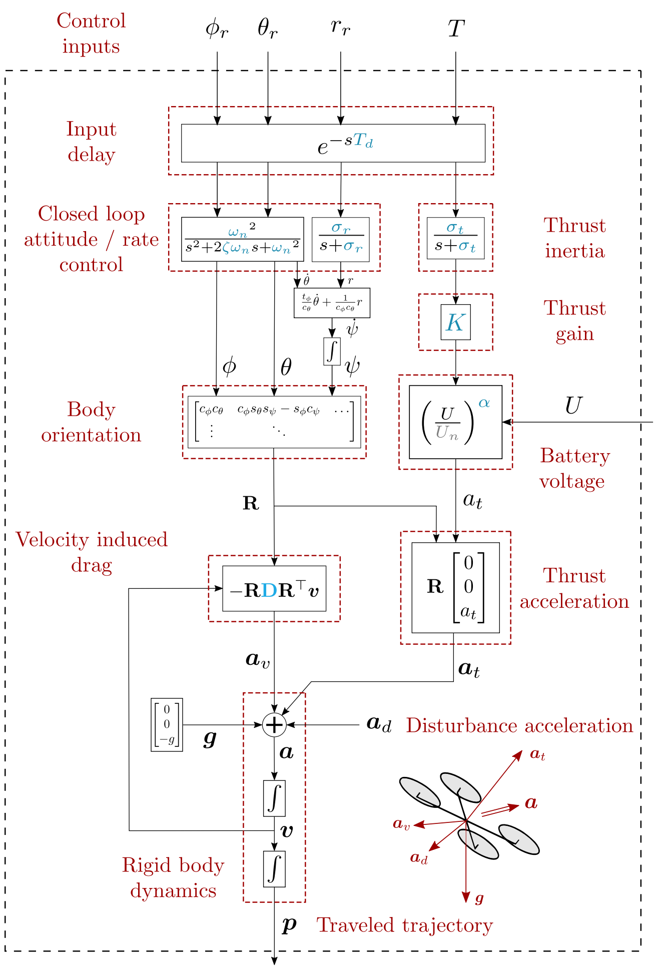

Control inputs
ϕ_r θ_r r_r T
Input delay
e^{-sT_d}
Closed loop attitude / rate control
$\frac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2}$
$\frac{\sigma_r}{s+\sigma_r}$
$\frac{\sigma_t}{s+\sigma_t}$
Thrust inertia
K
Thrust gain
$\frac{t_\phi}{c_\theta}\dot\theta + \frac{1}{c_\phi c_\theta} r$
$\dot\psi$ ψ ϕ θ
Body orientation
$\left(\frac{U}{U_n}\right)^\alpha$
U
Battery voltage
$\mathbf{R}$ a_t
Velocity induced drag
$-\mathbf{RDR}^\top \boldsymbol{v}$
Thrust acceleration
$\boldsymbol{a}_v$ $\boldsymbol{a}_t$
$\boldsymbol{g}$ $\boldsymbol{a}_d$ Disturbance acceleration
$\boldsymbol{a}$ $\boldsymbol{v}$
Rigid body dynamics
$\boldsymbol{p}$ Traveled trajectory

Figure 2: Schematic representation of the proposed model.

position or motion. The control inputs are the roll angle, pitch angle and yaw rate references ϕ_r, θ_r, r_r and the throttle input T. The communication and processing delay between the CC and the FCU is approximated by lumping it into an input delay. The responses of the attitude and thrust with respect to the delayed references are modeled as linear time-invariant second order systems for the roll and pitch, and first order systems for the yaw rate and thrust. The delayed and filtered throttle input is multiplied by a thrust gain and a battery voltage dependent gain, as the battery voltage affects the delivered thrust for a given throttle input. $\mathbf{R}$, the rotation matrix from the body to the world frame, is composed with the Euler angles ϕ, θ and ψ, following the axis convention displayed in Figure 1. It is not linearized, allowing for an accurate representation of large tilt angles. The orientation and the thrust acceleration magnitude a_t are prerequisites to obtain the rigid body dynamics.

The rigid body dynamics are considered as those of a point mass on which four forces act, represented by an equivalent acceleration through division by the drone mass: thrust acceleration $\boldsymbol{a}_t$, velocity induced drag acceleration $\boldsymbol{a}_v$, gravitational acceleration $\boldsymbol{g}$ and an unmodeled disturbance acceleration $\boldsymbol{a}_d$. The thrust acceleration and the drag acceleration closely follow the proposed model from Faessler et al. [6]. The thrust acceleration, corresponding to Faessler's mass-normalized collective thrust term, is assumed to be oriented along the body z-axis. Axial rotor drag causing thrust loss is not accounted for in the thrust acceleration, but instead is included in the drag acceleration. The drag acceleration follows Faessler's 'RDRv' model with $\mathbf{D}$ a constant diagonal matrix containing drag acceleration coefficients. Complex aerodynamic effects as introduced in Section 1 are neglected. The velocity and position are obtained through integration of the acceleration.

The equations of the resulting nonlinear state space model, which is of the form $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\Pi})$, are given in Equation 4. A list of all variables and constants is given in Table 1. The set of twelve parameters $\boldsymbol{\Pi}$ that fully characterize the system are highlighted in blue and are listed in Table 2. How to obtain their values is discussed in the next section. Overbarred symbols represent delayed inputs as in $\bar{u} = u(t - T_d)$. Vectors and their components are expressed in the inertial world frame.

$$\begin{bmatrix} \dot p_x \\ \dot p_y \\ \dot p_z \\ \dot v_x \\ \dot v_y \\ \dot v_z \\ \dot\phi \\ \dot\theta \\ \dot\psi \\ \ddot\phi \\ \ddot\theta \\ \dot r \\ \dot a_t \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ \dot v_x^{(*)} \\ \dot v_y^{(*)} \\ \dot v_z^{(*)} \\ \dot\phi \\ \dot\theta \\ \frac{t_\phi}{c_\theta}\dot\theta + \frac{1}{c_\phi c_\theta} r \\ -\omega_{n,\phi}{}^2\phi - 2\zeta_\phi\omega_{n,\phi}\dot\phi + \omega_{n,\phi}{}^2\bar\phi_r \\ -\omega_{n,\theta}{}^2\theta - 2\zeta_\theta\omega_{n,\theta}\dot\theta + \omega_{n,\theta}{}^2\bar\theta_r \\ -\sigma_r r + \sigma_r \bar r_r \\ -\sigma_t \boldsymbol{a}_t + K\left(\frac{U}{U_n}\right)^\alpha \sigma_t \bar T \end{bmatrix} \quad (4)$$

$$(*) \quad \underbrace{\begin{bmatrix} \dot v_x \\ \dot v_y \\ \dot v_z \end{bmatrix}}_{\dot{\boldsymbol{v}}} = \underbrace{\mathbf{R}\begin{bmatrix} 0 \\ 0 \\ \boldsymbol{a}_t \end{bmatrix}}_{\boldsymbol{a}_t} \underbrace{-\mathbf{RDR}^\top \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}}_{\boldsymbol{a}_v} - \underbrace{\begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}}_{\boldsymbol{g}}$$

In this equation the rotation matrix $\mathbf{R}$ from the body to the world frame in terms of the roll-pitch-yaw Euler angles is defined as was described in Section 1.

3 Parameter Estimation

To identify the parameters of the selected model, we follow a procedure solely based on flight data, unlike previous works describing cumbersome identification procedures on test benches or based on CAD models. This procedure broadens the applicability towards both open and closed source

Symbol	Variable	Physical units
T	Throttle input	-
ϕ	Roll Euler angle	rad
θ	Pitch Euler angle	rad
ψ	Yaw Euler angle	rad
ϕ_r	Roll angle reference	rad
θ_r	Pitch angle reference	rad
$\dot{\phi}$	Roll Euler rate	rad/s
$\dot{\theta}$	Pitch Euler rate	rad/s
$\dot{\psi}$	Yaw Euler rate	rad/s
p	Roll rate	rad/s
q	Pitch rate	rad/s
r	Yaw rate	rad/s
r_r	Yaw rate reference	rad/s
U	Battery voltage	V
$\mathbf{R}$	Rotation matrix	-
$\boldsymbol{a}$	Acceleration vector	m/s^2
a	Acceleration magnitude	m/s^2
$\boldsymbol{v}$	Translational velocity vector (ground speed)	m/s
$\boldsymbol{p}$	Position vector	m
$\boldsymbol{a}_t$	Thrust acceleration vector	m/s^2
a_t	Thrust acceleration magnitude	m/s^2
$\boldsymbol{a}_v$	Velocity induced drag acceleration vector	m/s^2
$\boldsymbol{a}_d$	Disturbance acceleration vector	m/s^2
$\boldsymbol{g}$	Gravitational acceleration vector	m/s^2
s	Laplace variable	s^{-1}
Symbol	**Constant**	**Value**
g	Gravitational constant	9.81 m/s^2
U_n	Nominal battery voltage	22.2 V

Table 1: Model variables and constants with their symbol and physical units.

Symbol	Parameter	Value
$\omega_{n,\phi}$	Roll natural frequency	25.43 rad/s
$\omega_{n,\theta}$	Pitch natural frequency	23.40 rad/s
ζ_ϕ	Roll damping ratio	0.44
ζ_θ	Pitch damping ratio	0.39
σ_r	Yaw rate decay constant	21.54 s^{-1}
σ_t	Thrust decay constant	20.36 s^{-1}
K	Thrust gain	54.55 m/s^2
α	Battery voltage exponent	1.71
d_x	Drag acceleration coefficient x	0.39 s^{-1}
d_y	Drag acceleration coefficient y	0.39 s^{-1}
d_z	Drag acceleration coefficient z	0.51 s^{-1}
$\mathbf{D}$	Drag acceleration coefficient matrix	$diag([d_x, d_y, d_z])$
T_d	Communication delay time	0.1 s

Table 2: Model parameters with their symbol and identified value.

platforms. Moreover, the procedure only uses sensor information that is available in relevant, outdoor scenarios.

To gather the data to estimate the parameters $\mathbf{\Pi}$, we perform three distinct experiments that each apply a maneuver repeatedly for varying input conditions, as illustrated in Figure 3. Decoupling the dynamics by applying step inputs on a subset of the inputs allows to assess the quality of the resulting fit more easily. The maneuvers can be executed within limited space, i.e. in this work specifically a flight zone of $30 \times 15 \times 10$ m shown in Figure 1. Each experiment results in M time series of varying length N. The sensors used by the FCU are 1) an integrated IMU [2], 2) a Here+ GPS module, 3) a lidar altitude rangefinder (Garmin LIDAR-Lite v3). As a consequence of the approximate modeling of closed loop subsystems, the full procedure including all maneuvers must be repeated as soon as one of the components (e.g. battery, propellers) of the drone setup changes.

Each of the maneuvers applies step inputs of varying magnitude. Firstly, the roll/pitch maneuver applies roll or pitch reference step inputs of 0.2 rad, 0.3 rad and 0.5 rad. The roll and pitch parameters $\mathbf{\Pi}_1 = [\omega_{n,\phi}, \zeta_\phi, \omega_{n,\theta}, \zeta_\theta]^\top$ are identified on the smaller steps of 0.2 rad, 0.3 rad to prevent angular rate saturation. Each of the steps is repeated three times to average measurement noise and random disturbances. Secondly, the yaw rate maneuver applies yaw rate reference step inputs of 1 rad/s and 2 rad/s, each repeated twice, to identify the yaw rate parameter $\mathbf{\Pi}_2 = \sigma_r$. Thirdly, the thrust maneuver applies varying throttle step inputs as illustrated in Figure 3 while recording the vertical acceleration and the battery voltage to gather data for ascending, descending and near to hovering conditions. The throttle input switches from T_0 at time t_0 to T_1 at t_1. Red arrows qualitatively indicate the magnitude of the applied throttle input. The thrust maneuver is executed for 18 combinations of T_0 and T_1, with T_0 ranging from 0.05 to 0.30 and T_1 ranging from 0.10 to 0.40, both in steps of 0.05. Each of the combinations is repeated three to six times over the battery voltage range from 21.5 V up to 24.5 V, resulting in a total of 72 recorded step responses. This serves to identify the thrust parameters $\mathbf{\Pi}_3 = [\sigma_t, K, \alpha]^\top$. The recorded orientation and acceleration in the roll/pitch maneuver allow to estimate the drag parameters $\mathbf{\Pi}_4 = [d_x, d_y, d_z]^\top$ after an initial fit of the thrust parameters. Only the data with roll/pitch reference steps of 0.3 rad and 0.5 rad is retained to assure sufficiently high acceleration such that the unmodeled disturbance acceleration is relatively low compared to the thrust acceleration and drag acceleration. For the same reason it is also important that the roll/pitch maneuvers are performed in near to windless conditions. Because the thrust identification is more accurate given accurate drag parameters and vice versa, their parameter estimates are updated iteratively until convergence.

With the gathered data, first, the delay time T_d is found

[2]LSM303D integrated accelerometer / magnetometer, L3GD20 gyroscope, MPU9250 Gyro / Accel

$\mathbf{\Pi}_j$	$e_{i,k}$	$\dot{\boldsymbol{x}} = \boldsymbol{f}_j(\boldsymbol{x}, \boldsymbol{u}, \Pi)$
$\mathbf{\Pi}_1$	$\widetilde{\phi}_{i,k} - \phi_{i,k}$	$\begin{bmatrix}\dot{\phi}\\ \ddot{\phi}\end{bmatrix} = \begin{bmatrix}0 & 1\\ -{\omega_{n,\phi}}^2 & -2\zeta_\phi\omega_{n,\phi}\end{bmatrix}\begin{bmatrix}\phi\\ \dot{\phi}\end{bmatrix} + \begin{bmatrix}0\\ {\omega_{n,\phi}}^2\end{bmatrix}\bar{\phi}$
	$\widetilde{\theta}_{i,k} - \theta i, k$	$\begin{bmatrix}\dot{\theta}\\ \ddot{\theta}\end{bmatrix} = \begin{bmatrix}0 & 1\\ -{\omega_{n,\theta}}^2 & -2\zeta_\theta\omega_{n,\theta}\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\end{bmatrix} + \begin{bmatrix}0\\ {\omega_{n,\theta}}^2\end{bmatrix}\bar{\theta}$
$\mathbf{\Pi}_2$	$\widetilde{r}_{i,k} - r_{i,k}$	$\dot{r} = -\sigma_r r + \sigma_r \bar{r}_r$
$\mathbf{\Pi}_3$	$\widetilde{a}_{z,i,k} - a_{z,i,k}$	$\begin{bmatrix}\dot{v}_z\\ \dot{a}_t\end{bmatrix} = \begin{bmatrix}a_t - D_z v_z - g\\ -\sigma_t a_t + K\left(\frac{U}{U_n}\right)^\alpha \sigma_t \bar{T}\end{bmatrix}$
$\mathbf{\Pi}_4$	$\Vert\widetilde{\boldsymbol{a}}_{i,k} - \boldsymbol{a}_{i,k}\Vert$	$\begin{bmatrix}\dot{v}_x\\ \dot{v}_y\\ \dot{v}_z\\ \dot{a}_t\end{bmatrix} = \begin{bmatrix}\mathbf{R}\begin{bmatrix}0\\0\\a_t\end{bmatrix} - \mathbf{R}\mathbf{D}\mathbf{R}^\top\boldsymbol{v} + \boldsymbol{g}\\ -\sigma_t a_t + K\left(\frac{U}{U_{nom}}\right)^\alpha \sigma_t \bar{T}\end{bmatrix}$

Table 3: Error term and evaluated dynamics for the estimation of each of the parameter subsets. The tilde indicates recorded data.

as the mean of the delay time between applied input and observed change on the corresponding output over all experiment data. Next, each of the remaining parameter subsets $\mathbf{\Pi}_j$ is estimated in a least squares optimization:

$$\begin{aligned}
\mathbf{\Pi}_j &= \underset{\mathbf{\Pi}_j}{\operatorname{argmin}}\, V \\
\text{subject to: } \boldsymbol{x}_{i,k+1} &= \boldsymbol{F}_j(\boldsymbol{x}_{i,k}, \bar{\boldsymbol{u}}_{i,k}, \mathbf{\Pi}_j) \quad \text{for } k=0,1,\dots,N_i-1, \\
& \qquad \text{for } i=1,2,\dots,M \\
\boldsymbol{x}_{i,0} &= \widetilde{\boldsymbol{x}}_{i,0}, \qquad \text{for } i=1,2,\dots,M
\end{aligned} \tag{5}$$

where

$$V = \sum_{i=1}^{M} V_i = \sum_{i=1}^{M}\sum_{k=0}^{N_i-1} e_{i,k}^2$$

$$\boldsymbol{F}_j(\boldsymbol{x}_k, \bar{\boldsymbol{u}}_k, \mathbf{\Pi}_j) = \int_{t_k}^{t_{k+1}} \boldsymbol{f}_j(\boldsymbol{x}_k, \bar{\boldsymbol{u}}_k, \mathbf{\Pi}_j)dt. \tag{6}$$

The integral from Equation 6 is evaluated using a fourth order Runge Kutta integration scheme. The error terms $e_{i,k}$ and the evaluated dynamics $\boldsymbol{f}_j$ for each of the identified parameter subsets are given in Table 3.

This least squares problem is solved for each of the parameter subsets using CasADi [13]. The resulting estimates for the parameters are given by Table 2. Figure 4 shows the measured roll, yaw rate and throttle input step responses together with the simulated results using the obtained parameters. The result for the pitch step responses is fully analogous to the roll step responses and is therefore not shown. The roll, yaw rate and throttle input step response data, on the top left, top right, and in the middle respectively, clearly show delayed predominantly second order and first order behavior, captured well by the model simulation. The middle plot shows the vertical acceleration as a function of time starting from t_1 and as a function of battery voltage for fifteen repetitions of the thrust maneuver, showing only throttle input combinations $T_0 = 0.05,\ 0.10,\ 0.15$ and $T_1 = 0.35$ for visual clarity. The acceleration magnitude, shown as a function

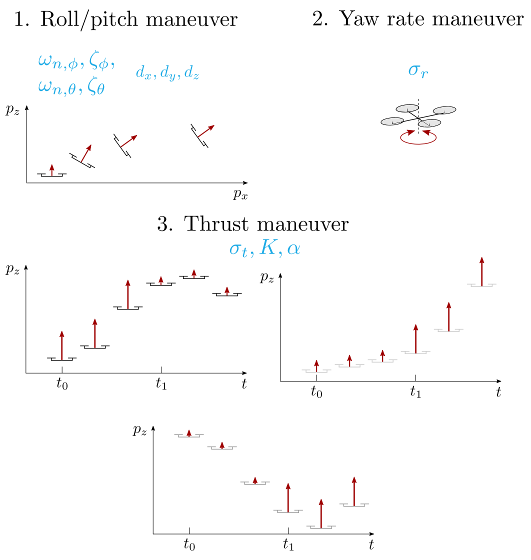


Figure 3: Illustration of identification experiment maneuvers, along with the parameters identified in the experiment.

of both time and velocity in the bottom plots, is predicted remarkably better with the inclusion of linear drag compared to the simulation without any drag. The arrows indicate the magnitude of the estimated drag acceleration.

4 Results and Discussion

To validate the prediction quality of the model and to motivate the inclusion of the modeled effects, its simulation on inputs from the first 1.5 s of a free flight using a rudimentary MPC controller is compared with the simulation of simplified versions of the model and the recorded experimental data. During this experiment, the battery voltage is around 23 V.

The models under comparison are 1) the full model as in Figure 2 and Equation 4, 2) a linearization of this model, treating yaw as described in [4], 3) the proposed model excluding drag, 4) the proposed model excluding the battery voltage dependency by assuming nominal battery voltage throughout the simulation. The roll, yaw rate and yaw responses are also compared to the situation where no first order or second order behavior is included and only the communication delay is considered. The adaptations to the full model to obtain the simplified versions are given by Table 4.

The simulation results in Figure 5, in which all plots correspond to the same short maneuver, prove that the full model predicts the drone motion significantly more accurately than the simplified versions of the model. Compared to the recorded data and the full model simulation, the only delay approximation clearly misses distinct evolutions of the attitude responses because it omits the first and second order behavior, as seen in the p, ϕ, r and ψ plots. The linear approximation strongly overestimates the vertical thrust component because of the assumption of small angles with respect to the

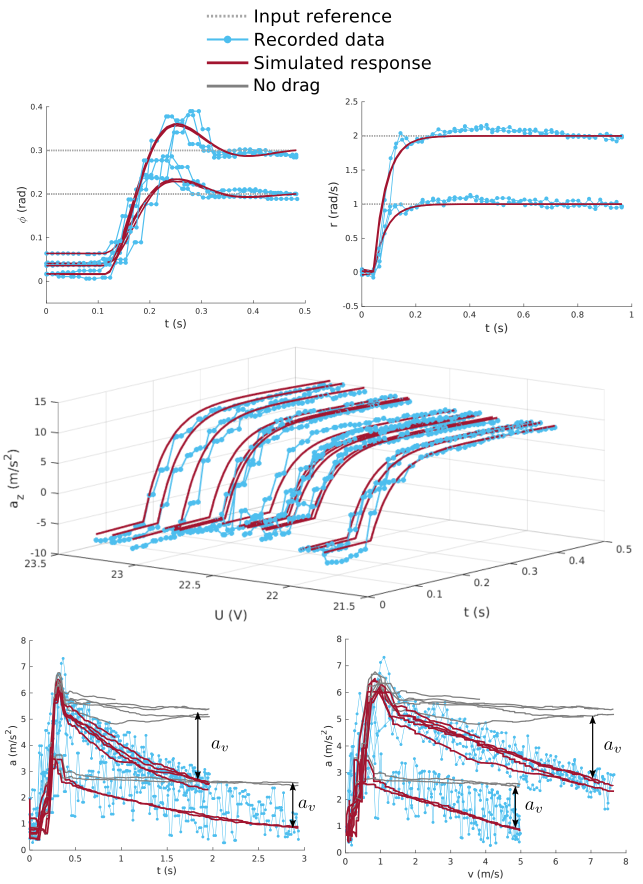


Figure 4: Validation of the model with estimated parameters.

Model variation	Model / change to the full model
Full model	$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\Pi})$
Linearized	$\delta\dot{\boldsymbol{x}} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\Big\vert_{\boldsymbol{x}^*,\boldsymbol{u}^*} \boldsymbol{x} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\Big\vert_{\boldsymbol{x}^*,\boldsymbol{u}^*} \boldsymbol{u}$, $\boldsymbol{x} = \boldsymbol{x}^* + \delta\boldsymbol{x}$, with $\boldsymbol{x}^*$ and $\boldsymbol{u}^*$ the hover state and hover inputs
No drag	$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\Pi})$ with $d_x = d_y = d_z = 0$
No battery dependency	$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\Pi})$ with $\alpha = 0$
Only delay	$\phi = \bar{\phi}_r, \theta = \bar{\theta}_r, r = \bar{r}_r$

Table 4: Variations of the full model for the comparison of prediction accuracy.

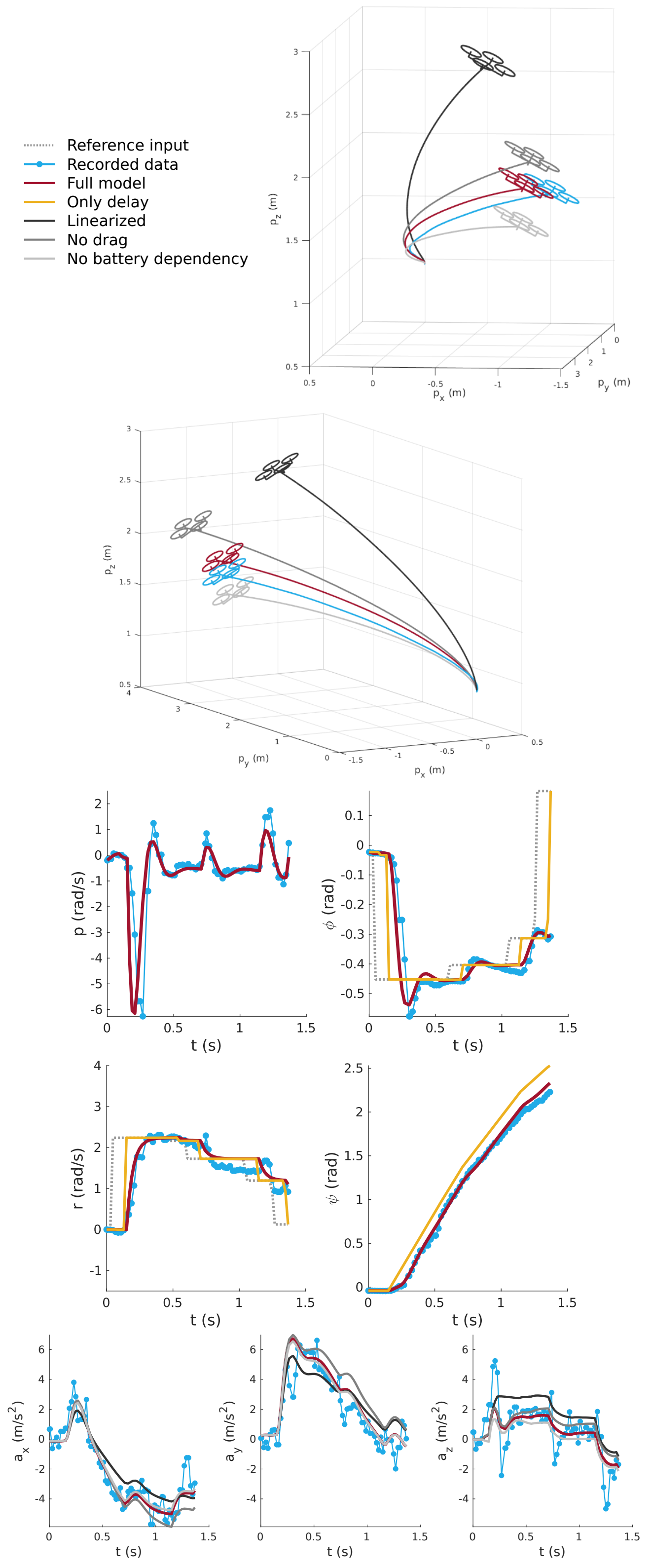


Figure 5: Comparison of four model variations of varying complexity with data recorded in the validation experiment. From top to bottom: 1) drone position, 2) roll rate and roll, yaw rate and yaw, 3) acceleration components in the world frame. All these plots correspond to the same short maneuver.

hover state, which is seen in the a_z and 3D position plots. The no drag approximation neglects the velocity induced counteracting force, hence overestimating the achieved acceleration, most clearly visible in the a_y and 3D position plots. The no battery dependency approximation neglects the effect of the higher battery voltage of 23 V compared to the nominal battery voltage of 22.2 V, hence underestimating the generated thrust and therefore acceleration. Consequently, the 3D position plots show a lower altitude for this simulation. These observations confirm the choice to include the low order approximation of low-level control loops, nonlinearity, drag effect and battery voltage dependency in the model.

To conclude this discussion, we discuss the limitations to the applicability of the proposed model that the approximations of the done dynamics made in this paper bring. Firstly, the low order decoupled approximation of control loops using an Euler angle representation and the assumption of linear drag restrict the use to low or medium velocity applications with limited roll and pitch angles. Secondly, saturation on the angular rates is not taken into account, meaning that for accurate predictions the high level planner should not request subsequent angle references that exceed the rate limits. Thirdly, simulating with attitude and throttle as inputs while neglecting a possible accelerometer bias in the identification procedure entails large drift on the velocity and position predictions over longer horizons, as the prediction error on the orientation and acceleration is integrated twice over time to obtain the position. This high level of uncertainty demands frequent corrections when using the model for estimation and control, which is not a problem in the intended MPC setting as this control strategy inherently comprises state feedback.

5 Conclusion and future work

This paper presented a multirotor drone model and a procedure to identify its unknown model parameters using the data measured during three simple maneuvers that do not require special facilities other than the sensors that are present in a realistic outdoor drone application and that do not require dismantling the drone to identify components separately. The model was crafted to address the trade-off between sufficient prediction accuracy and limited model complexity that arises when considering autonomous drone applications. The complexity of the representation of the low level control systems steering the attitude and thrust was reduced by modeling them as low order linear time-invariant subsystems. Linear drag and simplified battery voltage dependency were proven to benefit the prediction accuracy when included in the simulation.

In future work we will exploit this model in MPC, aiming for outdoor applications with a requirement for dynamic autonomous control. Early tests show approximate planning update rates of 20 Hz and more for time horizons over 10 s, which, possibly supported by (linear) feedback control, is adequate for many applications. Another interesting use of this model is the estimation of force disturbances, which were left untouched in this paper. Disturbance estimation could serve to estimate and reject wind influence for improved position tracking performance, and to estimate and compensate for the relative position and swinging of suspended payloads.

Acknowledgements

This work has been carried out within the framework of projects Flanders Make SBO MULTISYSLECO: Multi-System Learning Control, Flanders Make SBO MULTIROB: Rigorous approach for programming and optimal control of multirobot systems, and Flanders Make SBO HySLAM: A Hybrid SLAM approach for autonomous mobile systems. Flanders Make is the Flemish strategic research centre for the manufacturing industry. This work also benefits from project G0C4515N of the Research Foundation - Flanders (FWO - Flanders).

References

[1] H. Nguyen, M.S. Kamel, K. Alexis, and R. Siegwart. Model predictive control for micro aerial vehicles: A survey. arXiv 2011.11104, 11 2020.

[2] Xiaodong Zhang, Xiaoli Li, Kang Wang, and Yanjun Lu. A survey of modelling and identification of quadrotor robot. *Abstract and Applied Analysis*, 2014:320526, Oct 2014.

[3] Emil Fresk and George Nikolakopoulos. Full quaternion based attitude control for a quadrotor. In *2013 European Control Conference (ECC)*, pages 3864–3869, 2013.

[4] M. Kamel, M. Burri, and R. Siegwart. Linear vs nonlinear mpc for trajectory tracking applied to rotary wing micro aerial vehicles. arXiv 1611.09240, 2017.

[5] Gabriel Hoffmann, Haomiao Huang, Steven Waslander, and Claire Tomlin. *Quadrotor Helicopter Flight Dynamics and Control: Theory and Experiment*.

[6] M. Faessler, A. Franchi, and D. Scaramuzza. Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *IEEE Robotics and Automation Letters*, 3(2):620–626, 2018.

[7] S Li, C De Wagter, CC de Visser, QP Chu, and GCHE de Croon. In-flight model parameter and state estimation using gradient descent for high-speed flight. *International Journal of Micro Air Vehicles*, 11, 2019.

[8] P. Bouffard, A. Aswani, and C. Tomlin. Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results. In *2012 IEEE International Conference on Robotics and Automation*, pages 279–284, 2012.

[9] Bart Theys and Joris De Schutter. Virtual motor torque sensing for multirotor propulsion systems. *IEEE Robotics and Automation Letters*, 6(2):4149–4155, 2021.

[10] A. Chovancová, T. Fico, Ĺ. Chovanec, and P. Hubinsk. Mathematical modelling and parameter identification of quadrotor (a survey). *Procedia Engineering*, 96:172 – 181, 2014. Modelling of Mechanical and Mechatronic Systems.

[11] ArduPilot Development Team. Ardupilot project. `github.com/ArduPilot/ardupilot`.

[12] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza. Alphapilot: Autonomous drone racing. arXiv 2005.12813, 2020.

[13] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, Mar 2019.

Motion-based MAV Detection in GPS-denied Environments

Erik Vroon [*1], Jim Rojer [†2], and Guido C. H. E. de Croon [‡1]

[1]MAVLab, Control and Operations, Faculty of Aerospace Engineering, TU Delft, The Netherlands
[2]Netherlands Organisation for Applied Scientific Research (TNO), The Hague, The Netherlands

Abstract

Drones need to detect and localize each other if they are to collaborate in multi-robot teams or swarms. In this paper, a method based on dense optical flow (OF) is developed that detects dynamic objects. This is achieved by comparing the flow vectors with the direction to the Focus of Expansion (FoE) in the image plane. A simulation in AirSim is developed to validate this approach and to create a data set for motion-based dynamic object detection. This simulation includes ground-truth FoE, depth, OF and IMU data. The results show that our method performs well if the OF vector's magnitude is large enough and its angle is sufficiently different from those of static world points. We expect that the presented method will serve as a useful baseline for deep learning methods using dense optical flow as input.

1 Introduction

Nowadays, Micro Air Vehicles (MAVs) are becoming more and more common. Reasons for their popularity include their high maneuverability, vertical take-off capabilities and ability to perform tasks that humans cannot endure [1]. To further enhance the capabilities of MAVs and overcome the individual limitations of MAVs, swarms of MAVs were introduced. To enable the proper functioning of the swarm, sensing of the environment and the other MAVs is paramount. In particular, the relative locations of MAVs inside the swarm are needed for collision avoidance and swarm coordination [2]. The most basic and robust method of obtaining the locations of other MAVs, is by exchanging positions obtained from Global Navigation Satellite System (GNSS) signals. However, GNSS signals are not always available, for example when the signals are blocked, spoofed, jammed or distorted by multipath effects.

Computer vision is a promising alternative, because cameras are small/lightweight and provide a vast amount of information [2]. There are two main types of approaches solving the relative localization problem. The first is to create a shared map of the environment and have the MAVs exchange their location in this map. Simultaneous Localization and Mapping (SLAM) is a wide field of research that targets the first type [3]. The second type of relative localization focuses on the detection of the MAVs themselves. This process is often simplified by the use of physical devices called markers. These can be either infrared (see the work of Walter et al. [4]) or ultraviolet (see the work of Roberts et al. [5]) LEDs, or colored objects. Markers require specific hardware changes to the device, which may not always be desirable. Markerless detection represents a more difficult problem. Some methods quite successfully rely on stereo vision [6, 7]. Of course, for resource minimization, methods using a single camera are of interest. Currently, the main approach with a single camera is to employ deep neural networks that detect other MAVs in a single image [8, 9]. These neural networks show promising results, but it is not yet clear how well the trained networks can deal with cluttered backgrounds. Moreover, if the drone appearance or environment changes substantially with respect to the training set, retraining may be necessary.

In order to obtain a solution that does not depend on stereo vision or markers and that is more generic compared to appearance-based methods, it may be useful to use optical flow. Optical flow has multiple advantages over its alternative vision-based methods. Firstly, MAVs will possibly be detected in situations where they are barely distinguishable for the human eye due to background clutter. Additionally, optical flow based methods are less dependent on shapes and appearances of MAVs compared to other methods. Finally, optical flow can offer a larger maximum detection range compared to active markers.

Some papers incorporate motion into their appearance-based neural network, the so-called hybrid methods, such as the work by Yoshihashi et al. [10], where the temporal information improves the performance of the object detector in situations where there is little contrast between the background and

[*]Email address: erik.vroon22@gmail.com
[†]Email address: jim.rojer@tno.nl
[‡]Email address: g.c.h.e.decroon@tudelft.nl

foreground. Nonetheless, it is a ground-based method and uses static cameras, which is less challenging compared to the situation where the observer is moving.

To our knowledge, the research done by Li et al. [11] is the only work using purely optical flow without artificial neural networks for the detection of MAVs from a moving observer in the air. With their method, they were able to detect other MAVs, even when they were barely visible because of their size and the cluttered background. It has approximately the same detection accuracy (87%) as appearance-based neural networks applied to MAV detection (maximum accuracy of approximately 90%) [8, 12]. However, it is based on some assumptions. The method of Li et al. is based on a combination of background subtraction and Lucas-Kanade optical flow. The background subtraction process assumes that the tracked objects have a very different motion compared to a distant background, of which the motion is modelled with a homography transformation.

This paper focuses on motion-based object detection to detect MAVs from onboard a moving MAV in more general, 3D environments. Specifically, we present an optical-flow-based algorithm to detect dynamic objects in video feeds from a moving camera. This is done by comparing the flow vectors with the direction to the Focus of Expansion (FoE) in the image plane. This method is applied to simulations run in AirSim [13]. These simulations output ground-truth FoE, depth, optical flow and IMU data, which are valuable for the development and validation of motion-based object detection techniques. The proposed algorithm's image processing pipeline is mostly 'traditional', exploiting knowledge on the properties of the (derotated) optical flow field. We believe that eventually purely deep learning motion-based methods will achieve higher performance, but expect that the presented, completely comprehensible pipeline will be a useful benchmark method. Moreover, the results of our method show some of the challenges that will also be faced by deep learning methods, including difficult detection for small optical flow, flow directions similar to those of static world points, and the fact that other dynamic objects are not differentiated from MAVs in our current pipeline. On this last point, in this paper all moving objects are assumed to be MAVs, except for the clouds, which we detect with a deep neural network. To output only MAVs in an environment with other types of dynamic objects, the pipeline has to be extended to differentiate MAVs from other moving objects.

2 Detection method

The object detection method is illustrated in figure 2. First, the optical flow (OF) field is derotated using the rotation rates of the IMU. The location of the FoE is calculated using the derotated flow. FoE is the point where the translational flow is 0. This is the motion direction of the camera. All static points in the environment move away from the FoE. Points that are closer to the camera in terms of depth, have larger flow. Points that are further away from the FoE have larger flow as well. Dynamic objects may move in other directions. Then the associated flow vectors do not point away from the FoE. Unfortunately, they may move away from the FoE leading to flow that is similar to static objects. The angle κ between the vector pointing towards the FoE and the flow vector is calculated, as illustrated in figure 1. The larger κ, the more likely a pixel belongs to an object moving relative to the camera. In the following subsections, the individual steps of the method are explained in detail. All code used to reproduce this method and its results can be found publicly online[1].

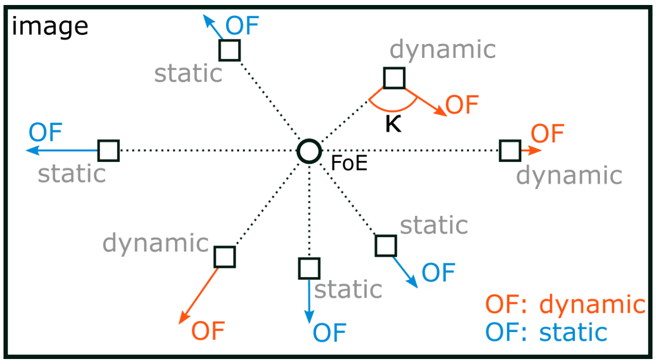


Figure 1: Illustration of κ for a camera moving forward. The κ angle denotes the difference between angle of the vector pointing at the FoE and the angle of the flow vector. For pixels of static objects, κ is approximately zero. For dynamic pixels, κ is non-zero, except when the object moves away from the FoE.

2.1 *Calculating optical flow*

As the object detection method relies on optical flow, an accurate dense optical flow estimator must be used. In figure 3, four neural networks estimating optical flow are compared. They illustrate that on the MIDGARD [14] dataset, LiteFlowNet [15] and Maskflownet [16] perform worse compared to RAFT [17] and FlowNet2 [18]. For all networks, the default weights were used. By visual inspection, FlowNet2 appears to perform best for small moving objects. Therefore, FlowNet2 is used for the results in the rest of this paper.

2.2 *Derotation*

Derotation has to be applied to the optical flow field to estimate an accurate FoE. The derotation technique in this paper is based on the work of Dinaux et al. [19]. The derotation vector per pixel coordinate can be calculated

[1]`https://github.com/evroon/mav-detection`

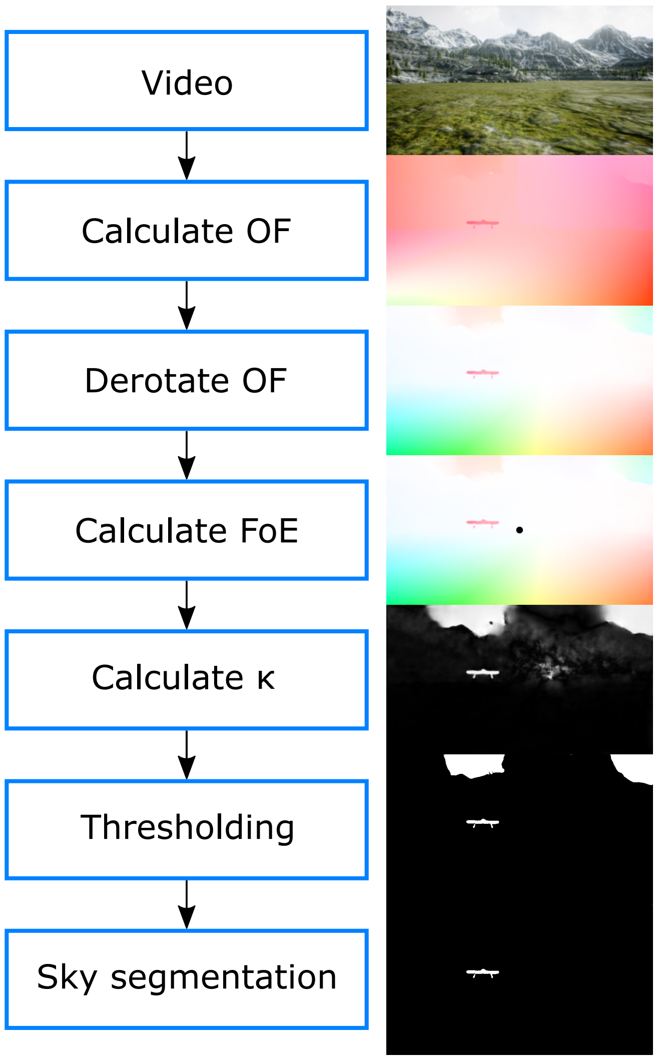
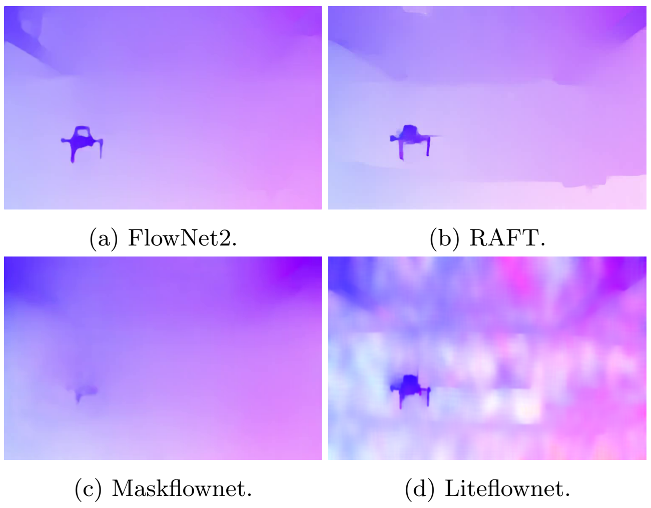


Figure 2: The proposed image processing pipeline.

from equation 1 describing optical flow (u, v) for a world point i in terms of ego-motion (U, V, W being the body velocities in X, Y, and Z direction and A, B, and C the rotations around those same axes) and the coordinates of the observed point (X_i, Y_i, Z_i, with image coordinates $x_i = \frac{X_i}{Z_i}$ and $y_i = \frac{Y_i}{Z_i}$), cf. [20].

$$u_i = -\frac{U}{Z_i} + x_i\frac{W}{Z_i} + Ax_iy_i - Bx_i^2 - B + Cy_i = u_T + u_R$$
$$v_i = -\frac{V}{Z_i} + y_i\frac{W}{Z_i} - Cx_i + A + Ay_i^2 - Bxy_i = v_T + v_R \tag{1}$$

The optical flow can be split into two factors: the rotational (u_R, v_R) and translational (u_T, v_T) parts. The rotational part is only dependent on the pixel coordinate and rotational rates of the camera (A, B, C). Therefore, the structure of the scene (in particular, depth) has no influence on the rotational part of optical flow. The Inertial Measurement Unit (IMU) of an MAV can be used to measure the rotational rate.



(a) FlowNet2. (b) RAFT. (c) Maskflownet. (d) Liteflownet.

Figure 3: Different neural networks estimating optical flow compared using the MIDGARD [14] dataset.

2.3 *Calculation of the FoE*

The Focus of Expansion (FoE) is the point where all flow vectors point towards or originate from when an observer moves through an environment. This point can lie outside the camera's Field of View, but in this paper it is assumed to lie in the image plane. Nonetheless, the method does work for FoEs outside the Field of View.

The FoE is calculated as presented in figure 4. First, two optical flow vectors are randomly sampled. The intersection of the two vectors is calculated. This process is repeated N times, where N equals 1000. A RANSAC scheme [21] is applied to the set of intersections to make it more robust against outliers. The RANSAC method calculates a location in the image where most intersections have a distance to this point that is lower than a certain threshold. The resulting location is taken as the location of the FoE.

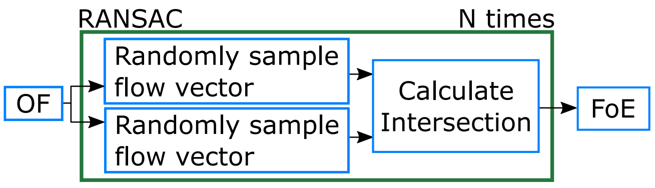


Figure 4: FoE method flowchart.

2.4 *Sky segmentation*

In outdoor environments, clouds in the sky can also move independently from the camera and generate substantial flow. Therefore, we segment clouds and sky by appearance and mask them out from the result. To this end, we use HRNet-OCR [22] with the default weights trained on the Cityscapes dataset [23]. By comparing the depth buffer from AirSim with the segmentation mask for the sky, one can validate the performance of the segmentation. Because of the visual simplicity of the environment in AirSim, the TPR of the sky segmentation is at least 99.5% and the FPR is less than 0.1%. The sky segmentation is performed at half the resolution of the captured images from AirSim, to reduce memory and computational effort of the GPU.

2.5 *Thresholding and detection output*

The output of the algorithm is based on the angle κ as illustrated in figure 1. The larger κ, the more likely it is that that pixel belongs to an object moving relative to the observer. Pixels with a κ angle larger than 15° are marked as moving objects. Out of these marked pixels, flow vectors with a magnitude smaller than 1 pixel/frame are discarded, because the angle of such vectors is sensitive to noise. However, the threshold on κ can be more substantiated by analyzing how the error in the angle of the flow vectors behaves for various magnitudes of flow. One would expect that the error of the estimated OF direction increases for decreasing OF magnitude. This is the case, as shown in figure 5. For 100 FlowNet2 images, the radial error with respect to the ground truth OF data is plotted for all pixels (except the sky) versus the magnitude of the OF. The white line of $0.25 \pm (0.5 + \frac{8}{|\text{OF}|})$ is fitted manually. The flow magnitude and value of κ that lie in the area between the upper and lower parts of this function, are discarded. Additionally, flow vectors with a magnitude lower than 0.5 pixels/frame are removed. The performance difference when using this 'dynamic' method of thresholding depending on the flow's magnitude is presented in the results section (see figure 10).

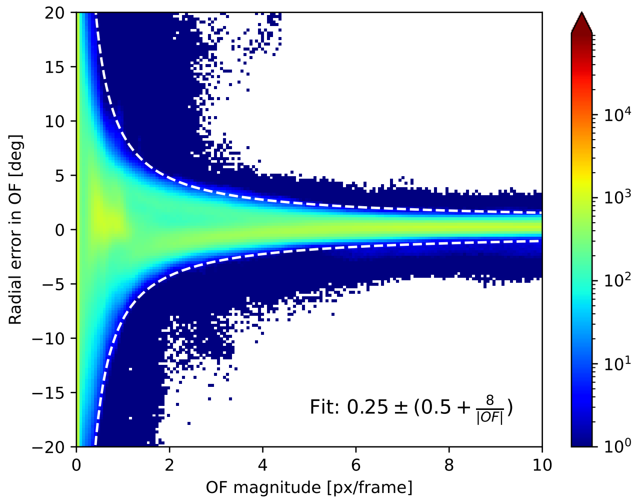


Figure 5: Histogram of the radial error in FlowNet2 (compared to the ground truth OF) versus the magnitude of the OF. Averaged over 100 OF fields.

3 AirSim

Simulations in AirSim [13] are carried out for various reasons. Most importantly, simulations can provide ground truth optical flow and FoE data that cannot be retrieved in real life. The ground truth optical flow makes it easier to develop a motion-based object detector, because the ground truth optical flow has no noise or artifacts. Simulations also enable validation of the algorithm on a low level, by for example comparing the FoE estimation with the ground truth FoE. Specifically, AirSim is chosen because of its realistic rendering and support for MAVs, including various simulated sensors.

3.1 *Environments*

One environment is used in AirSim: LandscapeMountains[2]. LandscapeMountains is a freely available project from Epic Games, the publisher of Unreal Engine. It was chosen because of its realism, while at the same time being not too demanding. To diminish the influence of visual effects on the estimation of optical flow and the performance of the object detector, most of these influences were removed from the simulation. All moving actors (gates to fly through, birds) are made invisible. The clouds are translated vertically by 500m such that they appear above the terrain. Additionally, to avoid reflections, the ice is replaced by a grass material and the fog is disabled. This limits the method to a set of real-world environments, but in a large range of applications these assumptions can still be considered valid. The only visible visual effect is the shadow of the terrain and MAVs.

3.2 *MAV control*

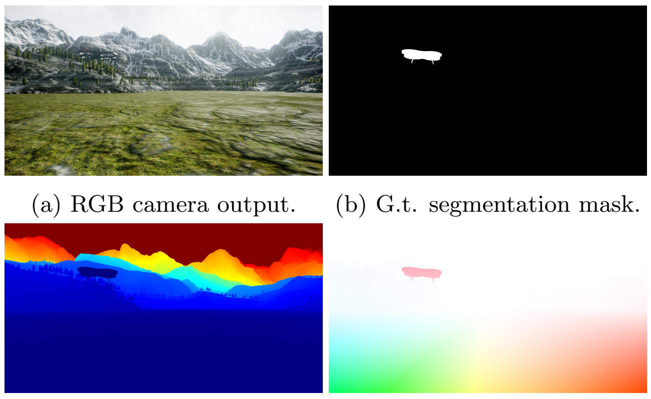
The MAVs are controlled using Python scripts. A loop is run for each simulation configuration, in which the MAVs are controlled and the data from AirSim is captured. First, the control inputs are calculated for the MAV to detect and the observing MAV. The time is advanced for 43ms (23Hz) and lastly, the data from AirSim is collected. The simulation is paused while obtaining the data of AirSim, such that the IMU data and camera frames are taken at the same timestep. The MAVs follow their flight path with a maximum deviation of 0.14m.

Two types of sequences are recorded. Firstly, collision courses, where the MAVs fly towards the same point at the same time at 4m/s. Secondly, sideways trajectories in which one MAV moves sideways in front of the observing MAV, which moves forwards at 4m/s.

[2]`https://www.unrealengine.com/marketplace/en-US/product/landscape-mountains`

3.3 *Data acquisition*

There are three visual outputs of the simulations: the RGB camera image, the depth in the camera image and the segmentation mask of the MAV inside the images. These three outputs are taken from the same camera, so all use the same projections. These outputs are shown in figures 6a to 6c. The camera image and segmentation mask are saved as PNG files, while the depth image is saved in AirSim's pfm format, enabling the use of floats. Additionally, sensor data is stored of both MAVs. This includes IMU and GPS data, but also contains collision data, the control inputs, FoE coordinates and camera properties. The ground truth FoE is calculated using the view projection matrix of the observer's camera and the observer's velocity vector. The images are collected at a resolution of 1920x1024 pixels with a framerate of approximately 23Hz. The field of view of the camera is 90° and there is no distortion or noise in the image.



(a) RGB camera output. (b) G.t. segmentation mask.

(c) Depth output. (d) Ground truth optical flow.

Figure 6: The different ground truth (g.t.) output frames captured in AirSim (a-c) and the g.t. optical flow (d) calculated from the depth output.

3.4 *Ground truth optical flow*

AirSim has no built-in method of calculating dense ground truth optical flow. However, it can be calculated from the depth image and the viewprojection matrix of the camera. This method is based on the work of Mayer et al. [24]. A visualization of the ground truth optical flow is shown in figure 6d and the steps of the method are shown in figure 7. Using the depth image, one can deduce the 3D world positions of all projected pixels by multiplying the inverse of the viewprojection matrix with the homogeneous pixel coordinates. This will result in a point cloud. From these 3D points, one can calculate their 3D positions one timestep ago. Finally, by applying the viewprojection matrix of the previous frame to the 3D points, one obtains the 2D coordinates of the original pixels one timestep ago. The difference between the original and the reprojected coordinates yields the ground truth OF. The optical flow calculation has some limitations. For example, the flow of visual effects is not taken into account. This includes shadows, animations of vegetation, reflections/refractions etc.

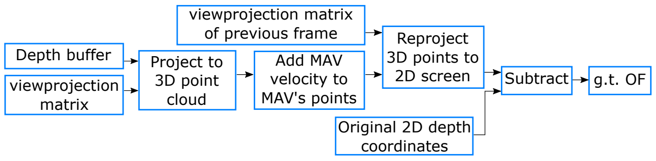

Depth buffer
viewprojection matrix
Project to 3D point cloud
viewprojection matrix of previous frame
Add MAV velocity to MAV's points
Reproject 3D points to 2D screen
Original 2D depth coordinates
Subtract
g.t. OF

Figure 7: Flowchart for calculating the ground truth OF.

3.5 *IMU*

The IMU is modeled using the default IMU in AirLib, the library implementing MAV dynamics and sensors inside AirSim. The biases and random walks of the gyroscope and accelerometer are set to zero, leaving IMU noise to future work. The IMU data is used for OF derotation, cf. subsection 2.2.

3.6 *Overview of parameters*

An overview of all parameters for the simulations and the object detection method is shown in table 1.

Table 1: Parameters of the simulation and method.

Parameter	Value
Resolution	1920x1024 px
Framerate	23 Hz
Field of View	90°
Observing MAV speed	4m/s
Fixed OF magnitude threshold	1 px/frame
Fixed OF radial threshold	15°
Number of collision course sequences	6
Number of sideways sequences	9
Number of FoE validation sequences	3

4 Results

This section will present the results in terms of performance on the FoE estimation and object detection for the simulations in AirSim.

4.1 *AirSim*

Because the accuracy of the object detection depends on the quality of the FoE estimation, the error between the estimated and ground-truth FoE is analyzed for different situations. A histogram of the FoE errors for one sequence is shown in figure 8. This is recorded for an MAV moving (without rotation) at 4 m/s with an FoE 20 pixels from the left and right edges of the image and an FoE in the center. Two characteristics are notable. For a forward moving MAV, the estimated FoE is on average slightly offset upwards (by 7.2 pixels) and to the right (by 2.8 pixels), but this is small compared to the total resolution of the image and therefore negligible for the majority of all pixels. Moreover, the location of the FoE affects the mean of the x distribution slightly, as the estimated FoE tends towards the center.

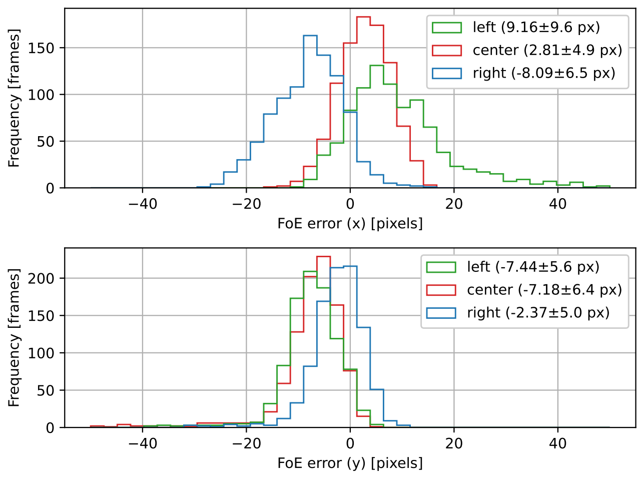


Figure 8: Histograms showing the error (in x and y direction) between the g.t. FoE and estimated FoE, for a FoE in the far left (at $x = 20$ px), center (at $x = 960$ px) and far right (at $x = 1900$ px) part of the image. The legend includes the means and standard deviations of the distributions.

The performance of the object detection method is determined by the True Positive Rate (TPR) and the False Positive Rate (FPR). TPR is the percentage of pixels from dynamic objects that are identified as dynamic object pixels. FPR is the percentage of pixels from static objects that are identified as dynamic object pixels. Ideally, one would have a large TPR for a very small FPR. In this case, the FPR is always relatively small, but the TPR varies considerably. This is shown in figure 9, where the TPR is plotted against κ for various speeds of the MAV to detect. As can be seen, the object detector is less accurate for slower moving objects.

The relation between the TPR/FPR and the magnitude of the OF of the detected object is presented in figure 10). The average TPR for κ between 180° and 90° is taken as measure of performance. It is clear that lower OF magnitudes decrease the TPR, but the FPR is unaffected. As hypothesized in section 2.5, a threshold that is dependent on the magnitude of the OF vector (a dynamic threshold) indeed results in a higher TPR for slower moving objects. However, this also increases the FPR to 0.5% - 2.0%, which could be considered acceptable depending on the application. In situations where the object to detect has a large OF vector, a fixed threshold would be more suitable.

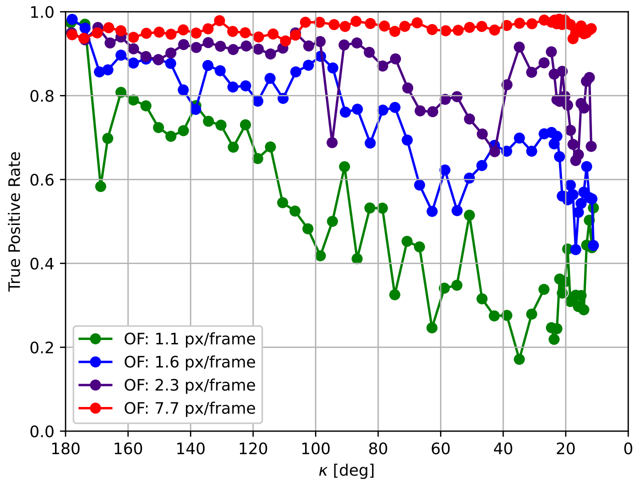


Figure 9: TPR vs κ, where MAV to detect moves from left to right with four different speeds (thus four magnitudes of OF) at a relative distance of 5m, decreasing κ from 180° to 0°. A dynamic threshold is applied.

Additionally, lower values of κ degrade the performance of the object detector. This is illustrated in figure 11, in which the angle κ is visualized. A higher intensity in the image indicates a higher value of κ, meaning that the flow vector is not pointing towards the FoE. Thus, such a flow vector does not only correspond to the flow created by the translation of the camera, but also to the motion of the object belonging to that pixel. In figure 11a, κ is large and therefore the MAV is easy to detect. In figure 11b, the MAV is more challenging to detect and in figure 11c, the method is unable to detect the MAV as κ is close to zero.

To test the method in more complex circumstances, data was recorded for a collision course where the flight paths of the MAVs cross at an angle of 75°, shown in figure 11d. In this case, the MAV to detect remains at the same location in the image during the sequence, but becomes closer and therefore larger in the image. It can be seen that the right part has a κ angle close to zero. However, using a dynamic threshold, the TPR is still high (0.98) at the cost of a relatively high FPR ($2.8 \cdot 10^{-2}$). Unfortunately, this is only the case for short distances. For a collision course, the flow magnitude at large distances is too small to properly estimate κ.

5 Discussion and Conclusion

We have introduced an optical-flow-based algorithm for detecting other moving objects, where our interest lies in the detection of other drones. The object detection method in this paper proves to work successfully if the angle of the optical flow vector of the object to detect is sufficiently different from the background flow, as

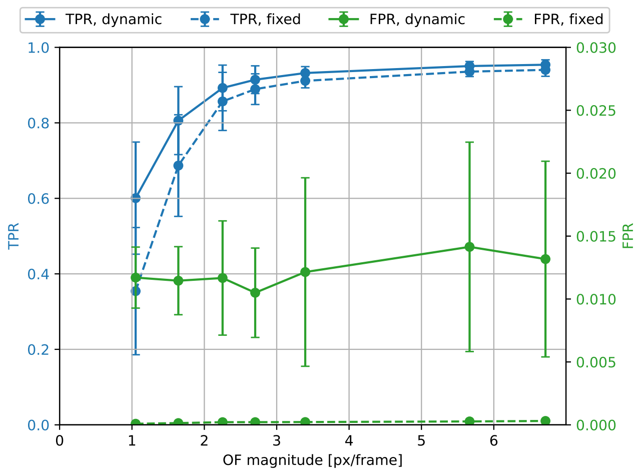
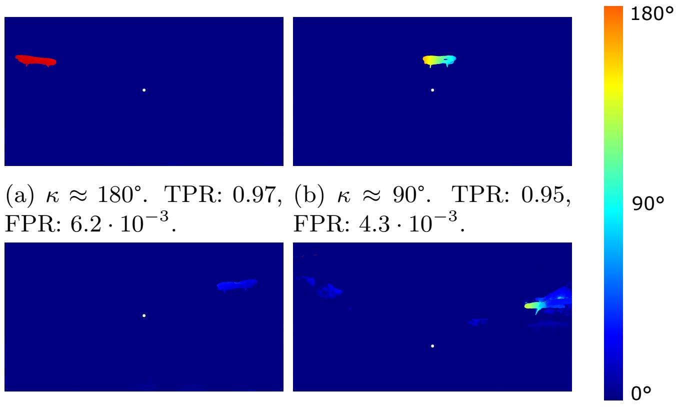


Figure 10: TPR and FPR vs the magnitude of the OF of the MAV to detect for $\kappa > 90°$ using a fixed and dynamic threshold.



(a) $\kappa \approx 180°$. TPR: 0.97, FPR: $6.2 \cdot 10^{-3}$. (b) $\kappa \approx 90°$. TPR: 0.95, FPR: $4.3 \cdot 10^{-3}$. (c) $\kappa \approx 0°$. TPR: 0.93, FPR: $1.5 \cdot 10^{-2}$. (d) CC. TPR: 0.98, FPR: $2.8 \cdot 10^{-2}$.

Figure 11: κ displayed for various situations. In (a) to (c), the MAV moves sideways from left to right. In (d), the observer and target are on a collision course (CC) of 75°. The white dot represents the FoE. A dynamic threshold is applied to calculate the TPR and FPR.

illustrated in figure 11. This means that objects moving towards the FoE, which are crossing the flight path of the observer and are thus considered dangerous, can be successfully detected. Although the method is based on assumptions of the OF, it does not assume a specific appearance of the moving object, which makes it suitable for a wide range of applications.

The method in this paper has the following limitations. Most importantly, if the observer is stationary or the dynamic object has no optical flow, detection by means of flow direction will not succeed. Therefore, MAVs on head-on collision courses cannot be detected in this way because they have the same flow field as the surroundings. A solution would be to utilize the divergence of the OF field to detect head-on colliding objects (just as for static objects). Another limitation is the computational effort of our current implementation, which is large due to the remaining deep network parts of the pipeline. For example, FlowNet2 runs on approximately 1.7 Hz on an RTX 2070 for 1920x1024 images. This would be too slow to use in real-time on MAVs themselves. Therefore, the resolution has to be reduced and/or another optical flow method must be used onboard.

References

[1] Shweta Gupte, Paul Infant Teenu Mohandas, and James M. Conrad. A survey of quadrotor unmanned aerial vehicles. *Conference Proceedings - IEEE SOUTHEASTCON*, 2012.

[2] Mario Coppola, Kimberly N. McGuire, Christophe De Wagter, and Guido C. H. E. de Croon. A Survey on Swarming With Micro Air Vehicles: Fundamental Challenges and Constraints. *Frontiers in Robotics and AI*, 7, feb 2020.

[3] Danping Zou, Ping Tan, and Wenxian Yu. Collaborative visual SLAM for multiple agents: A brief survey. *Virtual Reality & Intelligent Hardware*, 1(5):461–482, oct 2019.

[4] Viktor Walter, Nicolas Staub, Antonio Franchi, and Martin Saska. UVDAR System for Visual Relative Localization with Application to Leader-Follower Formations of Multirotor UAVs. *IEEE Robotics and Automation Letters*, 4(3):2637–2644, jul 2019.

[5] James F. Roberts, Timothy Stirling, Jean Christophe Zufferey, and Dario Floreano. 3-D relative positioning sensor for indoor flying robots. *Autonomous Robots*, 33(1-2):5–20, 2012.

[6] Matous Vrba, Daniel Hert, and Martin Saska. Onboard Marker-Less Detection and Localization of Non-Cooperating Drones for Their Safe Interception by an Autonomous Aerial System. *IEEE Robotics and Automation Letters*, 4(4):3402–3409, 2019.

[7] Changhong Fu, Adrian Carrio, Miguel A. Olivares-Mendez, Ramon Suarez-Fernandez, and Pascual Campoy. Robust real-time vision-based aircraft

Tracking from Unmanned Aerial Vehicles. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5441–5446. IEEE, may 2014.

[8] Bilal Taha and Abdulhadi Shoufan. Machine Learning-Based Drone Detection and Classification: State-of-the-Art in Research. *IEEE Access*, 7:138669–138682, 2019.

[9] Fabian Schilling, Julien Lecoeur, Fabrizio Schiano, and Dario Floreano. Learning Vision-Based Flight in Drone Swarms by Imitation. *IEEE Robotics and Automation Letters*, 4(4):4523–4530, oct 2019.

[10] Ryota Yoshihashi, Tu Tuan Trinh, Rei Kawakami, Shaodi You, Makoto Iida, and Takeshi Naemura. Differentiating objects by motion: Joint detection and tracking of small flying objects. *arXiv*, 2017.

[11] Jing Li, Dong Hye Ye, Timothy Chung, Mathias Kolsch, Juan Wachs, and Charles Bouman. Multi-target detection and tracking from a single camera in Unmanned Aerial Vehicles (UAVs). *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4992–4997, oct 2016.

[12] Cemal Aker and Sinan Kalkan. Using Deep Networks for Drone Detection. *arXiv*, jun 2017.

[13] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. *arXiv*, pages 1–14, may 2017.

[14] Viktor Walter, Matous Vrba, and Martin Saska. On training datasets for machine learning-based visual relative localization of micro-scale UAVs. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 10674–10680, 2020.

[15] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. A Lightweight Optical Flow CNN - Revisiting Data Fidelity and Regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020.

[16] Shengyu Zhao, Yilun Sheng, Yue Dong, Eric I.Chao Chang, and Yan Xu. Maskflownet: Asymmetric feature matching with learnable occlusion mask. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, c:6277–6286, 2020.

[17] Zachary Teed and Jia Deng. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12347 LNCS:402–419, 2020.

[18] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:1647–1655, 2017.

[19] Raoul Dinaux, Nikhil Wessendorp, Julien Dupeyroux, and Guido de Croon. FAITH: Fast iterative half-plane focus of expansion estimation using event-based optic flow. *arXiv*, feb 2021.

[20] H.C. Longuet and K. Prazdny. The Interpretation of a Moving Retinal Image. *Proceding of the royal society of london*, 208(1173):385–397, 1980.

[21] Martin A. Fischler and Robert C. Bolles. Random sample consensus. *Communications of the ACM*, 24(6):381–395, jun 1981.

[22] Andrew Tao, Karan Sapra, and Bryan Catanzaro. Hierarchical Multi-Scale Attention for Semantic Segmentation. *arXiv*, pages 1–11, may 2020.

[23] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:3213–3223, 2016.

[24] N Mayer, E Ilg, P Häusser, P Fischer, D Cremers, A Dosovitskiy, and T Brox. A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4040–4048, 2016.

Framework and evaluation methodology for Autonomous Drone Racing

Miguel Fernandez-Cortizas, Pablo Santamaria, David Perez-Saura, Javier Rodriguez-Vazquez, Martin Molina, Pascual Campoy

Computer Vision and Aerial Robotics group (CVAR), Centre for Automation and Robotics (CAR), Universidad Politécnica de Madrid (UPM-CSIC), Calle José Gutiérrez Abascal 2, 28006 Madrid, Spain.

Abstract

In recent years, autonomous drone races have become increasingly popular in the aerial robotics research community, due to the challenges in perception, localization, navigation, and control at high speeds, pushing forward the state of the art every year. However, autonomous racing drones are still far from reaching human pilot performance and a lot of research has to be done to accomplish that. In this work, a complete architecture system and an evaluation method for autonomous drone racing research, based on the open source framework Aerostack 4.0, are proposed. In order to evaluate the performance of the whole system and of each algorithm used separately, this framework is validated not only with simulated flights, but also through real flights in an indoor drone race circuit by using different configurations.

1 Introduction

1.1 Motivation

Autonomous drones have been increasing their application in different tasks in recent years. The short flight time of a quadcopter limits its use in missions such as search and rescue. To take advantage of this, it is interesting to develop agile drones that can explore or traverse an area in a short time. Autonomous drone racing is a great environment to push agile drones to the limit. The high speeds and agile maneuvers required for this purpose increase the difficulties of locating, controlling, and generating trajectories. To test different techniques to get the best results, it is useful to work in a modulated environment that allows you develop and validate different algorithms independently.

1.2 Related Work

Over the years, many different techniques have been developed to get the best results in Autonomous Drone Racing. At the beginning, the speeds achieved in competition were considerably low. In ADR 2017, a combination of monocular SLAM localization algorithm and a PID control won with a mean speed of 0.7 m/s [1]. For IROS 2018 ADR, the winners used machine learning techniques for gate detection with an MPC controller. They also improved the trajectories adding 2 points to the path, one before and one after each gate, getting flight speeds near 2 m/s [2].

In 2019, there was a big improvement in how fast autonomous drones could fly. For the first AlphaPilot competition, a novel architecture was developed [3]. In this architecture, machine learning techniques are combined with a nonlinear filter for sensor detection and time-optimal trajectory planning. Using a PD controller, they reached a maximum flight speed of 8 m/s. In the same year, the winners of the first simulated drone racing, NeurIPS 2019 Game of Drones, developed a controller using reinforcement learning techniques [4], achieving a maximum speed of 16 m/s in simulation.

On the other hand, there are many simulation environments to test the different approaches. Moreover, many competitions have been hosted in this simulators. Flightgoggles [5] is a photo-realistic simulator used for AlphaPilot 2019. Some of them have some APIs for a specific development. Flightmare [6] has an API for reinforcement learning. AirSim Drone Racing Lab [7] is a framework from Microsoft with some APIs that allows you to focus your test on each of the different research directions in autonomous drone racing. However, there are not many frameworks that combine state-of-the-art algorithms with simulators to work as baseline repositories for autonomous drone racing research.

1.3 Contribution

In this paper, we present a modular framework for developing and validating new algorithms for improving agile drone flying using autonomous drone races as a perfect test environment. This framework provides a modular system architecture with some state-of-the-Art algorithms that allows researchers to concentrate efforts on improving one of the fields related to drone behavior, without needing to build a whole system by themselves. Furthermore, a simulation environment based on gazebo is provided to test the algorithms before jumping into real experiments. For real experiments, we decided to use Pixhawk as the autopilot to ease the use of this framework through the research community.

Finally, we propose a set of metrics for measuring the performance of some modules separately and the performance of the whole system to be able to compare different algorithms easily.

2 System Architecture

The system architecture that is presented in this work has been developed using the Aerostack framework [8], an open-source multi-purpose software framework for the development of autonomous multi-robot unmanned aerial systems created by the Computer Vision and Aerial Robotics (CVAR) group. The Aerostack modules are mainly implemented in C++ and Python languages and are based on Robot Operating System (ROS) [9] for inter-communication between the different components, we refer the reader to the extensive documentation and publications that are available on its web site [1].

Using the Aerostack software framework, a new system architecture design was developed for the tasks presented in this work. It has to be noted that, although Aerostack framework is able to provide predefined components and intercommunication methods to provide autonomy to UAVs, the components that are described in this work were completely designed and developed for the objectives described in this paper. Fig. 1 shows the functionalities that have been implemented in this work. In the figure, colored rectangular boxes represent data processing units (or processes in short) that are implemented as ROS nodes. They are organized in the following main components:

- *Sensor-Actuator Interfaces:* to receive data from sensors on the aerial platform and send commands to robot actuators.
- *Communication Channel:* Based on the Aerostack framework, our architecture uses a common communication channel that contains shared dynamic information between processes. This channel facilitates process interoperability and helps to reuse components across different types of aerial platforms. The channel is implemented with a set of ROS topics and ROS messages.
- *Robot Behaviors:* Robot behaviors implement the robot functional capabilities including motion control, feature extraction, state estimation, and navigation.
- *Mission Control:* Mission control executes a mission plan specified in a formal description. In this implementation, the mission plan is specified using the Python language using a set of prefixed functions to start and stop robot tasks. A behavior coordinator component [10] is used to translate planned tasks into consistent activations of robot behaviors.

This work utilises the behaviorlib library for programming robot behaviors with execution management functions [10]. This library is open-source and provides tools for building, executing, and monitoring behaviors, as it is influenced by the behavior-based paradigm in robotics. According to this paradigm, the global control is divided into a set of behavior controllers and each one is in charge of a specific control aspect separately from the other behavior controllers.

Next sections describe in more detail the components related to robot behaviors and mission control.

3 Robot Behaviours

A behavior defines a basic functionality of a system, such as moving to a point, moving an actuator, activating a sensor, and includes the three following principles:

- *Common data channel:* Each behavior controller should be able to execute separately assuming that the required input data is available in the common data channel.
- *Activation management:* Each behavior is programmed with an activation management mechanism which handles how to start and stop the execution of the behavior controller.
- *Execution monitoring:* Execution monitoring is a kind of self-awareness computing process by which the robot observes and judges its own behavior. This includes possible behavior termination states such as ”GOAL ACHIEVED”, ”WRONG PROGRESS” or ”PROCESS FAILURE”.

Aerostack provides behaviors which can be divided in the categories explained in the following subsections.

3.1 Motion control

All the set of behaviors that are responsible for controlling the movement of the drone. This category includes simple behaviors such as take-off, hovering or landing, as well as more complex behaviors such as generating a trajectory and making the drone follow it.

In order to realize aggressive maneuvers on a quadrotor, it is necessary to employ a non-linearized controller. We have implemented a quadrotor control algorithm based on differential flatness and the corresponding behaviors inspired by the work made by Melligner et al. [11] with several modifications, so the output of the controller corresponds to angular velocity references and the desired collective thrust of all motors. The input of this controller consists in the position, speed and acceleration references provided by a trajectory generator.

Due to the need of generating trajectories for the controller, a polynomial trajectory generator [12][13][11] has been used. The trajectories are generated on a set of waypoints and are optimal in acceleration, which guarantees smoothness in the actuator commands. Moreover, the trajectories generated are constrained by maximum speed v_{max} and maximum acceleration a_{max} parameters.

[1] `www.aerostack.org`

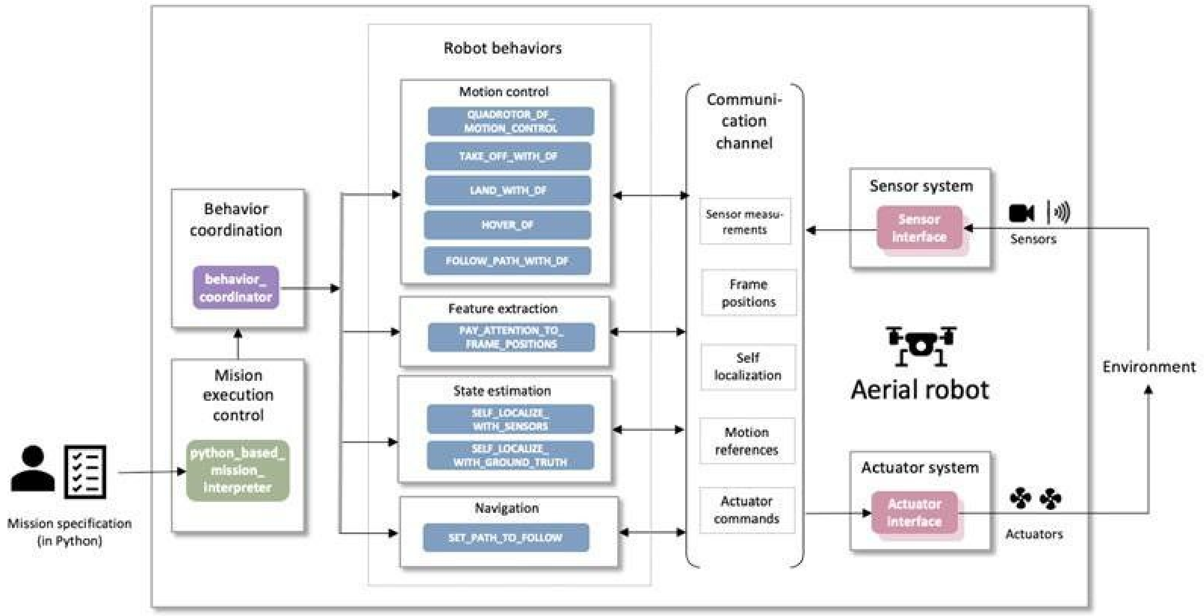


Figure 1: System architecture

3.2 *State estimation*

In order to achieve the level of control required for this project, it is necessary to provide the best state estimation possible to all components involved. Aerostack provides multiple components for this task, obtaining data from multiple sensors such as cameras (for relative positioning to a certain object), IMU, laser sensors, depth cameras, or lidar.

Some of this sensors or estimators should not be always trusted, as cumulative errors can and will happen, so it is necessary to combine the feedback from several of them using sensor fusion approaches, like the multi-sensor-fusion algorithm developed by Linnen et al. [14] , to achieve the most accurate and reliable state estimation possible in every situation.

However, for the real flights, we decided to use the state estimation provided by an Intel Realsense T265 Tracking Module, due to the ease of use and the reduction of the computational load of the on-board computer.

3.3 *Perception*

Perception is a key problem in the proposed scenario, since UAVs need to detect and locate each of the gates accurately to be able to complete the full track.

Since the detection and pose estimation of the gates is a difficult problem itself, to evaluate the rest of the proposed framework components without the influence of possible errors in perception, we placed ArUco fiducial markers [15] in each gate to estimate the relative position to the UAV's main camera. Each marker also encodes a unique gate ID, enabling us to design arbitrarily complex tracks. For the detection of such markers, we use OpenCV [16] implementation of the method proposed in [17]

3.4 *Navigation*

When the gates are located in the real world, it is necessary to plan the path that the aircraft must follow to pass through them and avoid other obstacles. For this approach we considered two scenarios:

- **Lack of knowledge of gate positions.** The aircraft does not have any information about where the gates are located, so it has to begin looking for them in order to generate the waypoints to pass through the gates. As long as the aircraft pass through the circuit, new gates will be in sight, so the quadrotor can add these gates to its route. In this scenario, the aircraft is always considered to see at least the following gate.

- **Gate position awareness.** The aircraft knows an approximated position of each gate of the circuit, so it can generate complete trajectories through all the circuit that must be corrected with as long as the gates are in sight, so it has to update the initial gate positions to pass through the circuit. This is how the majority of the autonomous drone racing competitions works.

In both approaches, each gate center is treated like a waypoint in a path, and this waypoint position is updated as the quadrotor flies through the circuit. Whenever the quadrotor passes through one gate this gate center is removed from the path. This path is sended to the trajectory generator, which takes charge of commanding the quadrotor to pass through the gates.

4 EXPERIMENTS

To validate the system architecture proposed and the suitability of the different algorithms selected, several experiments have been realized, not only in simulation, but also in real. For analyzing this performance, several metrics have been used:

- **State Estimation error:** For measuring the accuracy of the state estimation algorithm, we compute the RMSE (Root Mean Squared Error) between the estimated state and the ground truth.
- **Trajectory following error:** To evaluate the performance of the controller proposed, we decide to measure the trajectory following error, calculated with the RMSE between the trajectory sent by the trajectory generator and the real trajectory followed by the aircraft.
- **Speeds:** In autonomous racing, other metrics like the maximum speed reached, the medium speed, or the elapsed time to go through all the circuit must be taken into consideration.

All metrics obtained from the different experiments were obtained automatically using an evaluation script on the raw data recorded during the flights.

4.1 Simulated Flights

Initially, the system was validated in simulation using Gazebo [18] simulator and the *iris* drone as a simulated quadrotor. We generate a 5 gates circuit distributed along a 25 x 20 x 3.5 m area, see Fig. 2. The position of each gate was known with an uncertainty of 3 meters, which forces the system to recalculate the gates positions to avoid crashing into them.

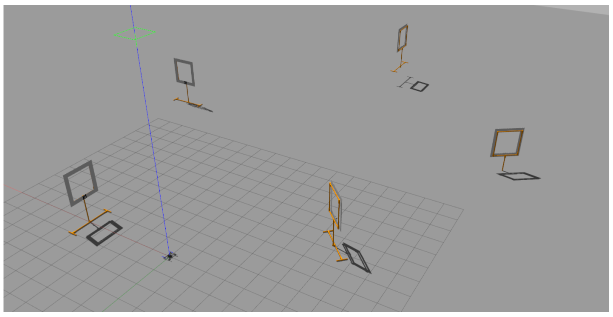


Figure 2: Gazebo environment used for simulated flight experiments

We fly through the circuit multiple times with four different speed configurations during each flight. In these experiments, the state of the aircraft was provided by the simulator, so the state estimation error was not calculated. All other metrics were obtained at different speeds, see Table 1 and Table 2.

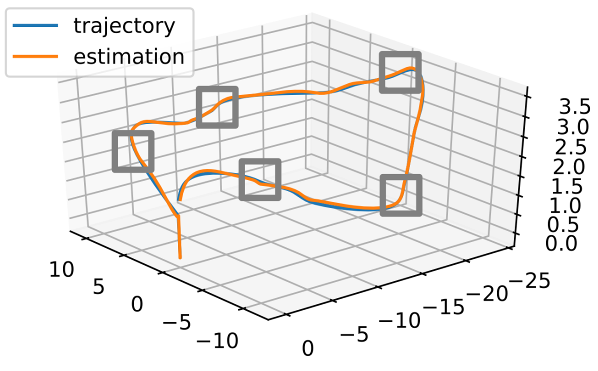


Figure 3: Path followed by the aircraft when passing through the simulated circuit with $v_{max} = 4.0 (m/s)$ compared to the trajectory generated for the motion control behaviour.

	x-axis	y-axis	z-axis	total
$v_{max} = 1.0$	0.0552	0.0572	0.0602	0.0807
$v_{max} = 2.0$	0.0647	0.0831	0.0734	0.1168
$v_{max} = 3.0$	0.0959	0.0963	0.0924	0.1468
$v_{max} = 4.0$	0.1001	0.1103	0.0952	0.1583

Table 1: RMSE between trajectory reference and estimator measurements, expressed in meters, when the simulated quadrotor pass through the whole circuit with trajectories generated with different values of v_{max} parameter

	V_{max}	V_{avg}	Elapsed Time (s)
$v_{max} = 1.0$	0.8192	0.3848	127.6
$v_{max} = 2.0$	1.5414	0.6896	61.1
$v_{max} = 3.0$	2.7687	1.1828	38.8
$v_{max} = 4.0$	3.2657	1.2243	34.5

Table 2: Speeds (m/s) achieve during flights and elapsed time to complete the whole simulated circuit employing different values of v_{max} parameter in the trajectory generation.

4.2 Aerial Vehicle Platform

The aerial platform used for the real experiments was a custom quadrotor based on the DJI F330 frame, shown in Fig. 4. This platform was equipped with a Pixhawk 4 mini as the aircraft autopilot, an Intel Realsense T265 Tracking Module used for state estimation, and an USB fish-eye camera for gate detection.

Additionally, the aerial platform was equipped with a Single Board Computer (SBC) NVIDIA Jetson Xavier NX with an 6-core ARM v8.2 , 64-bit CPU running Ubuntu Linux 18.04 Bionic Beaver for on-board computing. All computations required for the real experiments occurred in this SBC.

In order to obtain the ground truth position of the aircraft during some experiments, a Motion Capture System (*mocap*) was used. For localizing the aircraft inside *mocap* area, several IR markers has been attached to the platform.

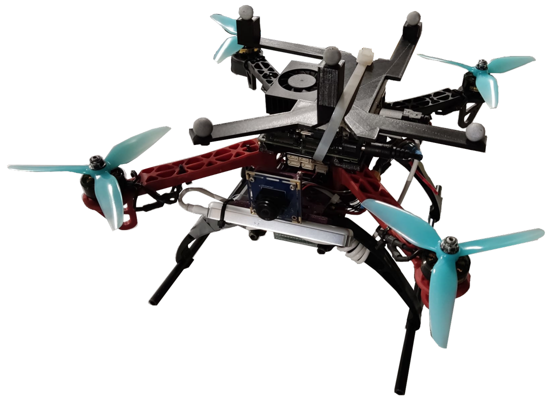


Figure 4: Quadrotor used for real flight experiments

4.3 *Real flights*

Due to space limitations in the *mocap* area we decided to do two different experiments: in the first one we make the drone pass through one gate with different speeds to evaluate the state estimation error and the trajectory following error of the controller, in the second one the aircraft had to pass through a small circuit with 4 gates to test the performance of the whole system in a complex task.

4.3.1 *One gate crossing*

For these experiments, the aircraft must localize a gate located in $gate_{position} = [2.0, 0.0, 1.5]$ (m) and pass through it with 4 different max speed $v_{max} = \{1.0, 2.0, 3.0, 4.0\}(m/s)$ values for the trajectory generation, see Fig. 5.

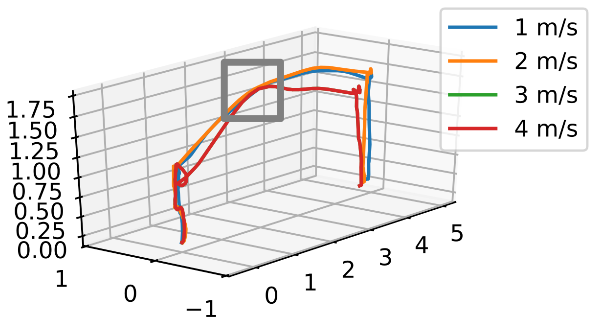


Figure 5: Path followed by the aircraft when passing through the gate at different speeds. The position measures had been obtained from the *mocap* system.

To measure the estimator error provided by the Realsense T265 Tracking Module when the quadrotor flies at different speeds we use *mocap* system for making a comparison between the ground truth and the estimated pose of the quadrotor, see Table 3.

	x-axis	y-axis	z-axis	total
$v_{max} = 1.0$	0.1140	0.0248	0.0990	0.1236
$v_{max} = 2.0$	0.1178	0.0173	0.1475	0.1561
$v_{max} = 3.0$	0.1079	0.0283	0.2313	0.2468
$v_{max} = 4.0$	0.1019	0.0545	0.3359	0.3474

Table 3: RMSE between Realsense estimation and ground truth measurements, expressed in meters, when the quadrotor flies through trajectories generated with different values of v_{max} parameter

Before flying through a more complex circuit, it is convenient to measure the trajectory following error of the controller employed when the aircraft flies at different speeds, see Table 4.

	x-axis	y-axis	z-axis	total
$v_{max} = 1.0$	0.0360	0.0262	0.0494	0.0558
$v_{max} = 2.0$	0.0683	0.0294	0.0633	0.0757
$v_{max} = 3.0$	0.1229	0.0385	0.0691	0.0948
$v_{max} = 4.0$	0.1699	0.0393	0.0820	0.0980

Table 4: RMSE between trajectory reference and estimator measurements, expressed in meters, when the quadrotor flies through trajectories generated with different values of v_{max} parameter

For evaluating the performance of the system passing through a drone racing circuit, other measures like the elapsed time to complete the circuit , the average flying speed, and the peak speed are needed, see Table 5.

	V_{max}	V_{avg}	Elapsed Time (s)
$v_{max} = 1.0$	0.9670	0.5625	9.4
$v_{max} = 2.0$	2.2556	0.9770	5.5
$v_{max} = 3.0$	3.1249	1.2059	4.7
$v_{max} = 4.0$	4.1673	1.5334	3.9

Table 5: Speeds (m/s) achieve during flights and elapsed time to complete the trajectory employing different values of v_{max} parameter in the trajectory generation.

4.3.2 *Four gates circuit crossing*

After validating the proper work of the whole system in the previous experiments, the last experiments consist in flying through a drone racing circuit with four gates arranged in the middle of the sides of a square of dimension 5x4 meters, with different heights each one.We fly through the circuit with two speed configurations: $v_{max} = 0.5$ and $v_{max} = 1.0$ as we can see in Fig 6 and Fig 7 respectively.

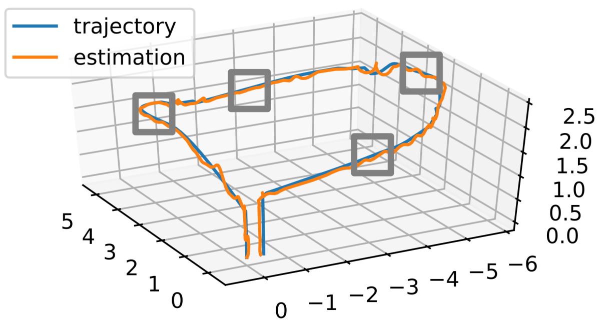


Figure 6: Path followed by the aircraft when passing through the 4 gates circuit with $v_{max} = 0.5(m/s)$ compared to the trajectory generated for the motion control behaviour.

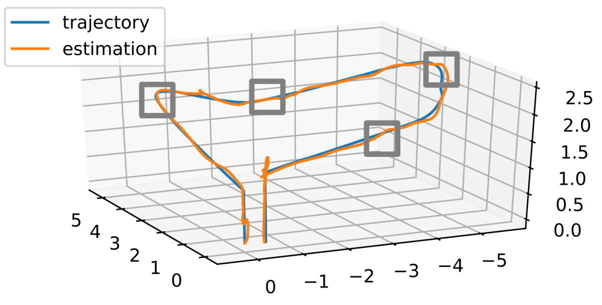


Figure 7: Path followed by the aircraft when passing through the 4 gates circuit with $v_{max} = 1.0(m/s)$ compared to the trajectory generated for the motion control behaviour.

Due to the space limitation of the arena, the ground truth poses were not acquired. However, the trajectory following errors of the controller and the time and speed metrics have been taken for comparing both flights, see Table 6 and Table 7.

	x-axis	y-axis	z-axis	total
$v_{max} = 0.5$	0.0492	0.0701	0.0576	0.0855
$v_{max} = 1.0$	0.0922	0.1491	0.1258	0.1414

Table 6: RMSE between trajectory reference and estimator measurements, expressed in meters, when the quadrotor pass through the whole circuit with trajectories generated with different values of v_{max} parameter

5 Results discussion

On the simulated flights, the system was able to complete the whole circuit at different speeds reaching speeds up to

	V_{max}	V_{avg}	Elapsed Time (s)
$v_{max} = 0.5$	0.5316	0.2060	79.1
$v_{max} = 1.0$	0.9231	0.4125	32.9

Table 7: Speeds (m/s) achieve during flights and elapsed time to complete the whole circuit employing different values of v_{max} parameter in the trajectory generation.

3.2 m/s, with the trajectory following average errors around 15 centimeters. which validates the operation of the system. Real experiments show that the proposed framework allows a real quadrotor to fly through drone racing circuit gates at speeds up to 4 m/s with a small increase in the trajectory following error compared with the simulated runs. The state estimator sensor can reach average estimation errors higher than 35 centimeters, adding this error to the trajectory following error could make the quadrotor collide with the gates if their positioning were not updated with respect to the drone position as long as the quadrotor navigates through the circuit. However, the limited computing capabilities of the SBC makes that the trajectory generation spends a lot of time, which worsens performance of the system when flying at high speeds through multiple gates.

6 Conclusions and Future Work

In this work, a modular framework for autonomous drone racing has been proposed and validated through several experiments, not only on simulation but also on real environments, being able to fly up to 4.16 m/s and to go through an small circuit successfully. The state-of-the-art algorithms proposed for each module, combined with the evaluation metrics proposed, constitute a baseline for research on improving autonomous agile drone flying.

To improve this framework, a wider range of possibilities to choose for each module must be given, like adding Predictive Model Controllers, learning-based gate estimation methods, or a Visual Inertial Odometry estimator fused with other sensors to improve the state estimation. Moreover, the domain gap between simulation and real life is substantial when non-photorealistic simulators are used. Using a simulator like Flightmare[6] would help to develop and test algorithms that rely on images taken through the flight.

Acknowledgements

This work has been supported by the project COMCISE RTI2018-100847-B-C21, funded by the Spanish Ministry of Science, Innovation and Universities (MCIU/AEI/FEDER, UE). The work of the first author is supported by the Research Assistant and Laboratory Technicians Program of the Science, University and Innovation Department of Madrid Government (partially funded by FEDER/EU) The work of the fourth author is supported Pre- and Post Doctoral Program of the Science, University and Innovation Department of Madrid Government (partially funded by FEDER/EU).

References

[1] Hyungpil Moon, Jose Martinez-Carranza, Titus Cieslewski, Matthias Faessler, Davide Falanga, Alessandro Simovic, Davide Scaramuzza, Shuo Li, Michael Ozo, Christophe De Wagter, Guido Croon, Sunyou Hwang, Sunggoo Jung, Hyunchul Shim, Haeryang Kim, Minhyuk Park, Tsz-Chiu Au, and si-jung Kim. Challenges and implemented technologies used in autonomous drone racing. *Intelligent Service Robotics*, 12, 04 2019.

[2] Elia Kaufmann, Mathias Gehrig, P. Foehn, René Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza. Beauty and the beast: Optimal methods meet learning for drone racing. *2019 International Conference on Robotics and Automation (ICRA)*, pages 690–696, 2019.

[3] Philipp Foehn, Dario Brescianini, Elia Kaufmann, Titus Cieslewski, Mathias Gehrig, Manasi Muglikar, and Davide Scaramuzza. Alphapilot: Autonomous drone racing. 05 2020.

[4] Y.-W. Kang S.-Y. Shin and Y.-G. Kim. Report for game of drones: A neurips 2019 competition.

[5] Winter Guerra, Ezra Tal, Varun Murali, Gilhyun Ryou, and Sertac Karaman. Flightgoggles: Photorealistic sensor simulation for perception-driven robotics using photogrammetry and virtual reality, 2019.

[6] Yunlong Song, Selim Naji, Elia Kaufmann, Antonio Loquercio, and Davide Scaramuzza. Flightmare: A flexible quadrotor simulator. 2020.

[7] Ratnesh Madaan, Nicholas Gyde, Sai Vemprala, Matthew Brown, Keiko Nagami, Tim Taubner, Eric Cristofalo, Davide Scaramuzza, Mac Schwager, and Ashish Kapoor. Airsim drone racing lab. *arXiv preprint arXiv:2003.05654*, 2020.

[8] Jose Luis Sanchez-Lopez, Martin Molina, Hriday Bavle, Carlos Sampedro, Ramon A Suarez Fernandez, and Pascual Campoy. A multi-layered component-based approach for the development of aerial robotic systems: The aerostack framework. *Journal of Intelligent & Robotic Systems*, 88(2), 2017.

[9] Stanford Artificial Intelligence Laboratory et al. Robotic operating system.

[10] Martin Molina and Pablo Santamaria. Behavior coordination for self-adaptive robots using constraint-based configuration. Arxiv preprint, 2021. arXiv:2103.13128.

[11] 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. IEEE, 2011.

[12] Charles Richter, Adam Bry, and Nicholas Roy. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In *Robotics Research*, pages 649–666. Springer, 2016.

[13] Michael Burri, Helen Oleynikova, , Markus W. Achtelik, and Roland Siegwart. Real-time visual-inertial mapping, re-localization and planning onboard mavs in unknown environments. In *Intelligent Robots and Systems (IROS 2015), 2015 IEEE/RSJ International Conference on*, Sept 2015.

[14] S Lynen, M Achtelik, S Weiss, M Chli, and R Siegwart. A robust and modular multi-sensor fusion approach applied to mav navigation. In *Proc. of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2013.

[15] Sergio Garrido-Jurado, Rafael Munoz-Salinas, Francisco José Madrid-Cuevas, and Rafael Medina-Carnicer. Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern Recognition*, 51:481–491, 2016.

[16] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[17] Francisco J Romero-Ramirez, Rafael Muñoz-Salinas, and Rafael Medina-Carnicer. Speeded up detection of squared fiducial markers. *Image and vision Computing*, 76:38–47, 2018.

[18] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3, 2004.

A Compact Approach for Emotional Assessment of Drone Pilots using BCI.

J. Daniel Olivares-Figueroa, Israel Cruz-Vega, J. M. Ramírez-Cortés, P. Gómez-Gil, and J. Martínez-Carranza
Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), Puebla, México

ABSTRACT

In this study, a method based on a *Brain-Computer Interface* (BCI) is proposed to continuously monitor emotional states related to the performance of drone pilots. As part of the contributions of this work, it is the creation of a database to classify two states: *Quiet* and *Very Tense*. The experiments were performed in a simulated environment. The EEG data of each participant was acquired using an EMOTIV Insight headset with 5 EEG channels. We propose an algorithm for automatic real-time artifact removal for five channels as a quick alternative. The *Asymmetry Index* (AI) is proposed as the main feature extracted from the frontal and temporal regions of the brain, followed by statistical measurements calculated from the AI vector to classify the signals with standard classifiers: *K-Nearest Neighbors* (KNN) and *Support Vector Machine* (SVM). We found clear evidence that the AI calculated in the frontal and temporal lobes of the brain is related to the response in drone pilots under emotional tension.

1 INTRODUCTION

Recently, drones for different applications such as civil and military service have increased, including maritime, space missions, search-rescue, shipping-delivery, etc. [1]. Despite being one of the most versatile tools, there are not enough studies that specifically focus on measuring the emotional state of drone pilots during the handling of unexpected emergencies.

Drone pilots engaged in long working hours manifest acute stress, which in the long term can turn into perceived stress [2], especially under adverse environmental conditions [3].

There are different methods for measuring human stress. Subjective methods use questionnaires [4], while objective methods use physical measures, for example facial expressions and blinking frequency, physiological processes, for example measuring the level of adrenaline in the blood, or using biosensors measuring the heart rate, brain waves, among others [2].

Neuroscience has shown that the response of human brain is affected by stress. Non-invasive technologies such as fMRI [5], [6], and *Electroencephalography* (EEG) [7] are the most common sources to study brain activity. However, EEG is a preferred application due to technological advances and commercial availability.

In this study, we propose a method based on a *Brain-Computer Interface* (BCI) to continuously monitor emotional states related to the performance of drone pilots, such as stress, fatigue, attention, and mental workload levels. We compute the *Asymmetry Index* (AI) [7] of the Alpha and Beta rhythms on frontal and temporal regions. The experiments were performed in a simulated environment under controlled conditions, obtaining eight statistical measurements to characterize the AI vector: *mean*, *median*, *standard deviation*, *RMS*, *peak-to-RMS*, *peak-to-peak*, *mean frequency*, and *power*. The proposed system employs these characteristics to train two classifiers: *K-Nearest Neighbors* (KNN) [8] and *Support Vector Machine* (SVM) [9]. The performance is evaluated using the average of *accuracy*, *precision*, *sensitivity*, and *specificity* [10].

To assess our model, a database was generated, which is divided into three classes: *Quiet*, *Tense*, and *Very Tense*. For the experiments presented here, we select the *Quiet* and *Very Tense* groups for the classification process.

The rest of this paper is structured as follows: Section 2 presents the relevant related work to this project. The database generation and its processing are presented in Sections 3 and 4, respectively. Results are shown in Section 5, and finally, the conclusion are presented in Section 6.

2 RELATED WORK

Emotions have a strong correlation with the left and right frontal lobes activity. Stronger activation in the left lobe is related to positive emotions. Instead, when the activation of the right lobe is relatively more significant, it represents mainly negative emotions [11], [2].

Numerous studies show clear evidence that frontal asymmetry is related to emotional responses and

*Email address(es): daniel_olfi@inaoep.com disorders. Theoretical background can be consulted in reference [12].

Studies related to the detection of stress suggest that frontal asymmetry is a promising biomarker. In [13], a method for identification of chronic stress is presented, finding that the average AI of the stressed group was lower than the group in relaxed condition. A similar result was found by [14], [15], where alpha and beta power asymmetry were analyzed. Low beta waves were analyzed in [4] to quantify human stress, using a single frontal channel efficiently. In [7], shows that the average alpha, beta, and gamma wave AI tends to be lower in the stressed group than in the controlled group and suggests that alpha asymmetry is the best candidate. The calculation of AI on alpha and beta regions reported in this work is based on [7].

3 DATABASE CREATION

Three healthy male volunteers participated in this experiment, with ages between 18 to 23 years old. One subject showed a high level of skills for video games, and the remaining two showed a moderate level; none of the participants had experience in handling drones. Data were acquired between 3 and 6 pm by the three participants.

The EEG data of each participant was acquired with an EMOTIV Insight headset with 5 EEG electrodes (AF3, AF4, T7, T8, Pz) and two reference electrodes (CMS/DRL) located in the left mastoid bone. A data transmission rate of 128 samples per second was used, with a passband of 0.5 to 43 Hz and a notch filter at 50 and 60 Hz (https://www.emotiv.com/insight/).

A database was generated with information obtained from the subjects in three emotional states: *Quiet*, *Tense*, and *Very Tense*. In addition, *Quiet* and *Very Tense* conditions have been used for classification.

3.1 Complementary Information

For each participant, we collect name, age, gender, experience level with video games or drone driving, time of experience, and relevant medical conditions such as injuries, surgeries, chronic diseases, and allergies.

3.2 Experimental Development Environment

In the experiment, we employed two screens, shown in figure 1. First, the operator controlled the experiment using a Graphical User Interface (GUI), label in the figure as "first screen," which is linked to an application provided by EMOTIV Insight developers, called EmotivPRO. Then, using the GUI labeled in figure 1 as "second screen," the participant fulfilled the tasks assigned on each test.

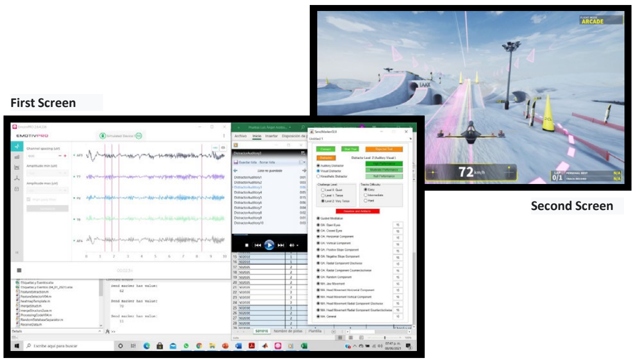


Figure 1: Experimental development environment. Screen for the operator (left) and for the participant (right).

3.3 Practice with the Flight Simulator

A preliminary training session allowed each participant to become familiar with the drone flight simulator and the control commands.

3.4 Recording Calibration Signals

Signals correlated with noise generated by different artifacts, as well as a baseline, were measured. Each subject listened a guided meditation audio for 5 minutes to induce a state of relaxation. Subsequently, each subject followed the instructions shown on the screen to measure ocular and muscular artifacts (eyes open, eyes closed, and movements in all directions of the jaw, neck, and eyes).

3.5 Experimental Tests

Each participant completed different challenges in the "*DCL the Game*" flight simulator (https://dcl.aero/), such as following trajectories and overcoming obstacles on each runway, to test precision and concentration skills. The tracks sizes range from 30 sec to 2 min, depending on the circuit and the pilot's skill in each test. Each session was applied on different days lasting from 25 min to 35 min and exposing the participants gradually to three stress levels: *Quiet*, *Tense*, and *Very Tense* (see Figure 2).

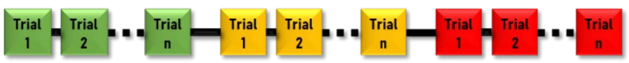


Figure 2: General scheme of a session. Green, yellow, and red blocks for *Quiet*, *Tense*, and *Very Tense* states, respectively. The solid lines joining each block correspond to a 30-seconds break.

Level 0 (Quiet): The participant performs basic maneuvers, such as taking off, landing, turning right, left, moving forward, and backward without obstacles.
Level 1 (Tense): The participant must run each track with obstacles without suffering an accident with the drone. The subjects are instructed to complete the tracks trying to beat his own record in time. The signals obtained from

each track are recorded and stored in the database. In addition, the subject was immersed in music of action and related genres to induce a more significant engagement.

Level 2 (Very Tense): The participant must fulfill the same tasks as Level 2, while being distracted with *auditory* and *visual* stimuli. *Auditory distractors* consisted of sudden, short-lived audios. *Visual distractors* consisted of randomly appearing images, blocking partial vision at different sizes and positions on the screen.

Signals have been labeled and organized according to the following characteristics: Emotional Tension Level, Track Difficulty, Distractors, Pilot Performance, and Test Start/End. In turn, each characteristic can assume one of three possible levels.

4 PROCESSING

Signals classification must perform as a real-time application. Therefore, there is a trade-off between efficiency and speed throughout the entire process. Figures 3, 4, and 5 show the algorithm proposed for real-time signal processing: *Detrend and Artifact Removal*, *Brain Rhythm Filter*, *Asymmetry Index Calculation*, *Feature Extraction*, *Model Training and Testing*.

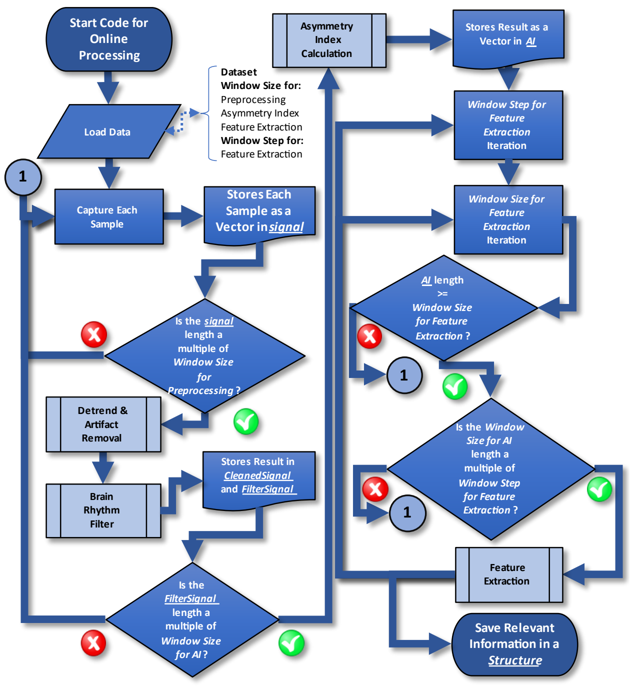

Start Code for Online Processing
Load Data
Dataset
Window Size for:
Preprocessing
Asymmetry Index
Feature Extraction
Window Step for:
Feature Extraction
1
Capture Each Sample
Stores Each Sample as a Vector in signal
Is the signal length a multiple of Window Size for Preprocessing ?
Detrend & Artifact Removal
Brain Rhythm Filter
Stores Result in CleanedSignal and FilterSignal
Is the FilterSignal length a multiple of Window Size for AI ?
Asymmetry Index Calculation
Stores Result as a Vector in AI
Window Step for Feature Extraction Iteration
Window Size for Feature Extraction Iteration
AI length >= Window Size for Feature Extraction ?
1
Is the Window Size for AI length a multiple of Window Step for Feature Extraction ?
1
Feature Extraction
Save Relevant Information in a Structure

Figure 3: Flowchart proposed for real-time signal processing.

4.1 *Detrend and Artifact Removal*

To detrend and remove the DC component of each EEG channel, we applied *Empirical Mode Decomposition* (EMD) [16], reconstructing the signals by omitting the three lowest frequencies, using the *Intrinsic Mode Functions* (IMF).

Several studies that show sophisticated techniques for artifact removal are limited to offline systems [17], [18]. On the other hand, studies that eliminate artifacts in real-time use 32 or 64 channels [19], focusing on eliminating a single artifact [20], [21], or using a reference signal [22].

We propose an algorithm, inspired in the works of [23] and [19], for automatic real-time artifact removal using five channels. Our aim is to avoid complex approaches proposing a practical and quickly applied alternative. Figure 4 shows the general process.

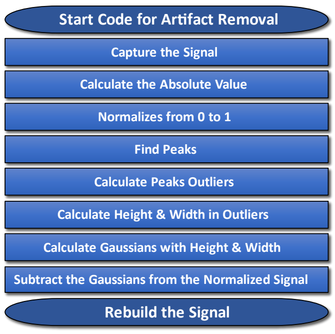

Start Code for Artifact Removal
Capture the Signal
Calculate the Absolute Value
Normalizes from 0 to 1
Find Peaks
Calculate Peaks Outliers
Calculate Height & Width in Outliers
Calculate Gaussians with Height & Width
Subtract the Gaussians from the Normalized Signal
Rebuild the Signal

Figure 4: Proposed algorithm for automatic removal of artifacts in real-time.

4.2 *Brain Rhythm Filter*

We designed two cascade IIR filters (high pass filter - low pass filter) Chebyshev Type II of minimum order to extract the Alpha (7-12 Hz) and Beta (12-30 Hz) rhythm. Both were applied using an attenuation in the rejection band of 60 dB per decade.

4.3 *Asymmetry Index Calculation*

The Alpha and Beta *Asymmetry Index* (AI) were calculated using equations (1), (2) and (3), as described in [7] and [24], and stored as a vector. Where AI_f, AI_t and AI, represent frontal, temporal, and total asymmetry, respectively. P_{AF4}, P_{AF3}, P_{T8} and P_{T7} represent the power of the corresponding channel for the rhythm of interest.

$$AI_f = \frac{P_{AF4} - P_{AF3}}{P_{AF4} + P_{AF3}} \tag{1}$$

$$AI_t = \frac{P_{T8} - P_{T7}}{P_{T8} + P_{T7}} \tag{2}$$

$$AI = AI_f + AI_t \tag{3}$$

4.4 *Feature Extraction*

Eight features (*mean*, *median*, *standard deviation*, *RMS*, *peak-to-RMS*, *peak-to-peak*, *mean frequency* and *power*) have been extracted from the asymmetry vector AI_f, AI_t y AI, giving a total of 24 for each rhythm.

4.5 *Model Training and Testing*

Several models are trained for each step and window size. In each execution (*Run Number*) the training set (80%) and test set (20%) are randomly selected, and different hyper-parameters were tested depending on the type of classifier. For the KNN classifier, distance type (*euclidean*, *seuclidean*, *cityblock*, *minkowski*, *chebyshev*, *cosine*, *correlation*, *spearman*) and K number of nearest neighbors (from 3 to 10) were iterated. For SVM, we used Kernel functions *linear*, *quadratic*, *cubic polynomial*, and *Gaussian* with a Kernel scale of 1.2, 4.9, and 20.

For each execution and iteration of hyper-parameters, models are evaluated in their training and testing stage with *accuracy*, *precision*, *sensitivity*, and *specificity* [10]. Finally, the mean of the four metrics was calculated to select the KNN and SVM hyperparameters with the best average for each step and window size (see figure 5).

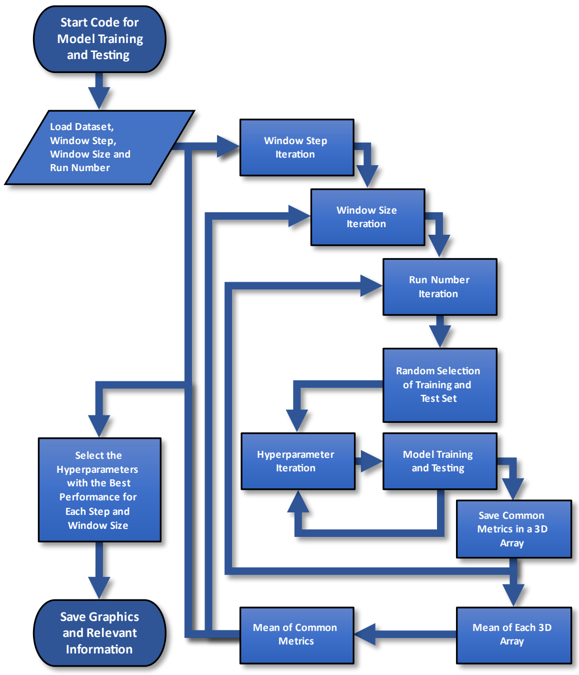

Start Code for Model Training and Testing
Load Dataset, Window Step, Window Size and Run Number
Window Step Iteration
Window Size Iteration
Run Number Iteration
Random Selection of Training and Test Set
Hyperparameter Iteration
Model Training and Testing
Save Common Metrics in a 3D Array
Mean of Each 3D Array
Mean of Common Metrics
Select the Hyperparameters with the Best Performance for Each Step and Window Size
Save Graphics and Relevant Information

Figure 5: Process for training and testing all models.

5 RESULTS AND DISCUSSIONS

The dataset used in the process described in figure 3 contains the data obtained from the tests applied to each subject. Preprocessing the signal and the calculation the AI were performed using a window of 2 seconds. Four window sizes of 30, 60, 90 and 120 s were considered to extract features. Three window shifts for feature extraction (see figure 3) were tested: 10, 20 and 30 s. The process shown in Figure 5 for the evaluation of the models during training and testing stage was repeated 20 times (*Run Number = 20*). Figures 6, 7 and 8 show the results obtained for each combination of step and window size. These correspond to the best average (in percentage) obtained from the four metrics (*accuracy*, *precision*, *sensitivity,* and *specificity*) at the test stage.

It is observed for all subjects that, regardless of the window step considered, the results tend to improve as the window size increases. The best performance for all three subjects is typically obtained when the window step is 10 s for both classifiers. We can see that the results are very similar among the classifiers.

Comparing the alpha and beta rhythms of the different steps and window sizes for each subject separately, we generally observe that the beta rhythm is higher than the alpha rhythm.

Tables 1 and 2 show the best results and the hyperparameters calculated in both classifiers for alpha and beta rhythms, respectively. We observe that for the KNN classifier, the best results are obtained using 3, 4 and 5 nearest neighbors. This favors the real-time application objective of our study, since the smaller the number of nearest neighbors, the shorter the time required for classification.

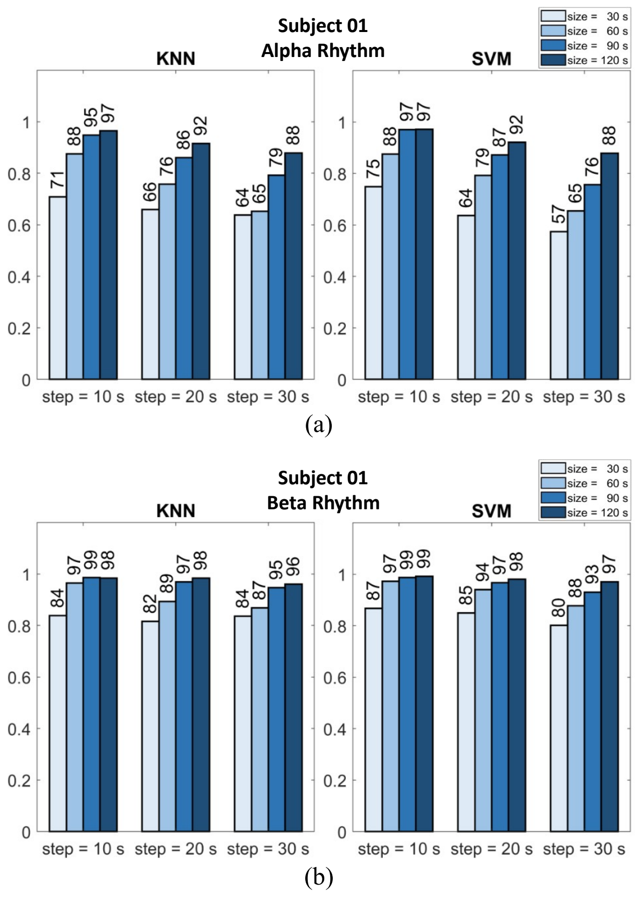
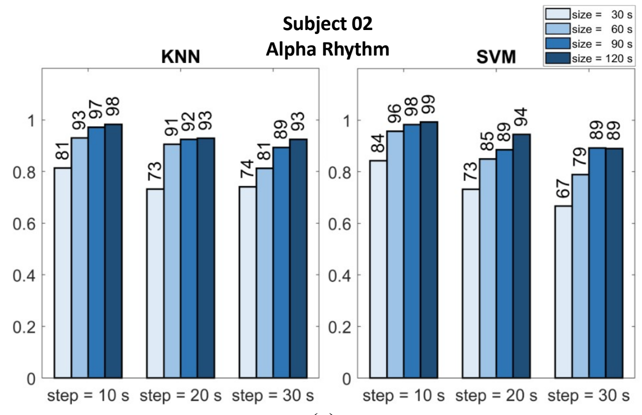
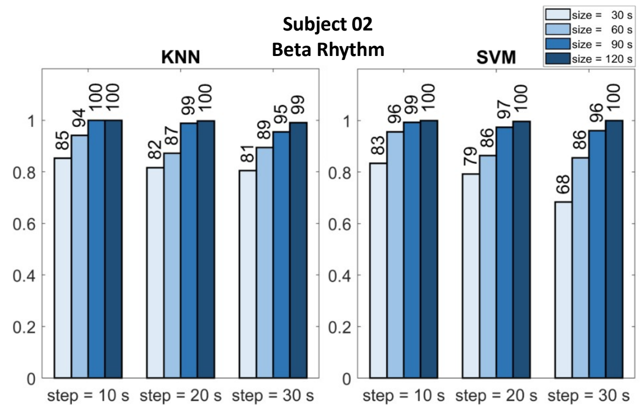


Figure 6: Subject 01. Mean of the metrics for the KNN and SVM classifiers of the (a) Alpha and (b) Beta rhythms.

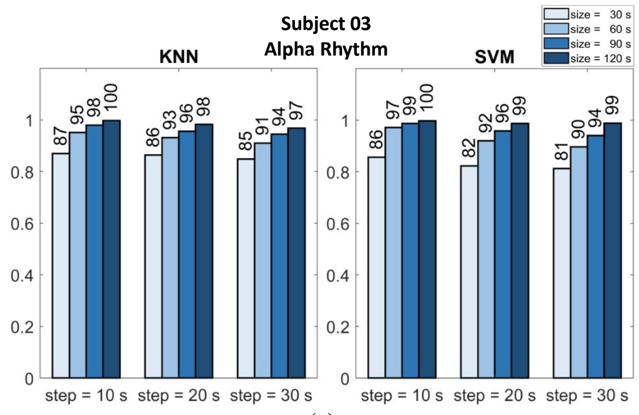


(a)

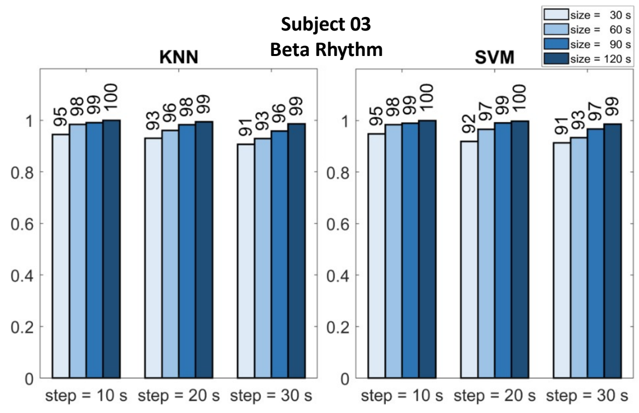


(b)

Figure 7: Subject 02. Mean of the metrics for the KNN and SVM classifiers of the (a) Alpha and (b) Beta rhythms.



(a)



(b)

Figure 8: Subject 03. Mean of the metrics for the KNN and SVM classifiers of the (a) Alpha and (b) Beta rhythms.

Alpha Rhythm

	Window		KNN		SVM	
	Step	Size	Metrics Average (%)	Hyper - parameters	Metrics Average (%)	Hyper - parameters
Subject 01	10	90			97	Polynomial Cubic
	10	120	97	Seuclidean K = 4	97	Polynomial Cubic
Subject 02	10	120	98	Cityblock K = 3	99	Polynomial Cubic
Subject 03	10	120	100	Cityblock K = 3	100	Polynomial Cubic

Table 1: Best results obtained for the alpha rhythm.

Beta Rhythm

	Window		KNN		SVM	
	Step	Size	Metrics Average (%)	Hyper - parameters	Metrics Average	Hyper - parameters
Subject 01	10	90	99	Euclidean K = 3	99	Fine Gaussian
	10	120			99	Polynomial Quadratic
Subject 02	10	90	100	Cityblock K = 3		
	10	120	100	Minkowski K = 5	100	Polynomial Quadratic
	20	120	100	Seuclidean K = 5	100	Medium Gaussian
	30	120			100	Polynomial Cubic
Subject 03	10	120	100	Minkowski K = 3	100	Polynomial Quadratic
	20	120			100	Polynomial Quadratic

Table 2: Best results obtained for the beta rhythm.

6 CONCLUSIONS AND FUTURE WORK

In this study, a method is proposed using a BCI to continuously monitor emotional states, which is related to the performance of drone pilots. We built a database to obtain three emotional states where *Quiet* and *Very Tense* states were classified using KNN and SVM. Our findings show that there is a clear separability between these two groups. We proposed an algorithm for automatic real-time artifact removal for five channels as a fast alternative.

We found that the AI in the Alpha and Beta waves is an excellent feature related to the emotional response in drone pilots in situations of emotional tension. Our study suggests that the results corresponding to the four metrics reported in figures 6, 7 and 8 indicate a better performance when the beta rhythm is used, in comparison to those obtained from the alpha rhythm.

Our next step is to expand the database, to test the generalization ability of our model. This database will be publicly available. Also, we will explore other classifiers techniques such as neural networks based on *Deep Learning*.

REFERENCES

[1] M. Hassanalian and A. Abdelkefi, “Classifications, applications, and design challenges of drones: A review,” *Prog. Aerosp. Sci.*, vol. 91, no. April, pp. 99–131, 2017, doi: 10.1016/j.paerosci.2017.04.003.

[2] A. Arsalan, M. Majid, A. R. Butt, and S. M. Anwar, “Classification of Perceived Mental Stress Using A Commercially Available EEG Headband,” *IEEE J. Biomed. Heal. Informatics*, vol. 23, no. 6, pp. 2257–2264, 2019, doi: 10.1109/JBHI.2019.2926407.

[3] A. Valenzano *et al.*, “Stress profile in remotely piloted aircraft crewmembers during 2 h operating mission,” *Front. Physiol.*, vol. 9, no. MAY, pp. 1–6, 2018, doi: 10.3389/fphys.2018.00461.

[4] S. M. U. Saeed, S. M. Anwar, and M. Majid, “Quantification of human stress using commercially available single channel EEG Headset,” *IEICE Trans. Inf. Syst.*, vol. E100D, no. 9, pp. 2241–2244, 2017, doi: 10.1587/transinf.2016EDL8248.

[5] K. S. Hong, M. J. Khan, and M. J. Hong, “Feature Extraction and Classification Methods for Hybrid fNIRS-EEG Brain-Computer Interfaces,” *Front. Hum. Neurosci.*, vol. 12, no. June, pp. 1–25, 2018, doi: 10.3389/fnhum.2018.00246.

[6] F. Dehais *et al.*, “Monitoring Pilot’s Cognitive Fatigue with Engagement Features in Simulated and Actual Flight Conditions Using an Hybrid fNIRS-EEG Passive BCI,” *Proc. - 2018 IEEE Int. Conf. Syst. Man, Cybern. SMC 2018*, pp. 544–549, 2019, doi: 10.1109/SMC.2018.00102.

[7] S. M. U. Saeed, S. M. Anwar, H. Khalid, M. Majid, and U. Bagci, “EEG based classification of long-term stress using psychological labeling,” *Sensors (Switzerland)*, vol. 20, no. 7, pp. 1–15, 2020, doi: 10.3390/s20071886.

[8] P. E. H. T.M. COVER, “Nearest Neighbor Pattern Classfication,” vol. I, pp. 1–28, 2012.

[9] A. Ben-Hur, C. S. Ong, S. Sonnenburg, B. Schölkopf, and G. Rätsch, “Support vector machines and kernels for computational biology,” *PLoS Comput. Biol.*, vol. 4, no. 10, 2008, doi: 10.1371/journal.pcbi.1000173.

[10] T. Fawcett, “An introduction to ROC analysis,” *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, 2006, doi: 10.1016/j.patrec.2005.10.010.

[11] M. Mohammadpour, S. M. R. Hashemi, and N. Houshmand, “Classification of EEG-based emotion for BCI applications,” *7th Conf. Artif. Intell. Robot. IRANOPEN 2017*, pp. 127–131, 2017, doi: 10.1109/RIOS.2017.7956455.

[12] J. A. Coan and J. J. B. Allen, “Frontal EEG asymmetry as a moderator and mediator of emotion,” *Biol. Psychol.*, vol. 67, no. 1–2, pp. 7–50, 2004, doi: 10.1016/j.biopsycho.2004.03.002.

[13] H. Peng *et al.*, “A method of identifying chronic stress by EEG,” *Pers. Ubiquitous Comput.*, vol. 17, no. 7, pp. 1341–1347, 2013, doi: 10.1007/s00779-012-0593-3.

[14] J. W. Ahn, Y. Ku, and H. C. Kim, “A novel wearable EEG and ECG recording system for stress assessment,” *Sensors (Switzerland)*, vol. 19, no. 9, 2019, doi: 10.3390/s19091991.

[15] F. Al-Shargie, M. Kiguchi, N. Badruddin, S. C. Dass, A. F. M. Hani, and T. B. Tang, “Mental stress assessment using simultaneous measurement of EEG and fNIRS,” *Biomed. Opt. Express*, vol. 7, no. 10, p. 3882, 2016, doi: 10.1364/boe.7.003882.

[16] N. E. Huang *et al.*, “The empirical mode decomposition and the Hubert spectrum for nonlinear and non-stationary time series analysis,” *Proc. R. Soc. A Math. Phys. Eng. Sci.*, vol. 454, no. 1971, pp. 903–995, 1998, doi: 10.1098/rspa.1998.0193.

[17] A. Santillan-Guzman, U. Heute, M. Muthuraman, U. Stephani, and A. Galka, “DBS artifact suppression using a time-frequency domain filter,” *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. EMBS*, pp. 4815–4818, 2013, doi: 10.1109/EMBC.2013.6610625.

[18] A. Santillán-Guzmán, U. Heute, U. Stephani, H. Mühle, A. Galka, and M. Siniatchkin, “Hybrid filter for removing power-supply artifacts from EEG signals,” *Proc. IASTED Int. Conf. Biomed. Eng. BioMed 2013*, no. BioMed, pp. 41–45, 2013, doi: 10.2316/P.2013.791-022.

[19] S. Çınar and N. Acır, “A novel system for automatic removal of ocular artefacts in EEG by using outlier detection methods and independent component analysis,” *Expert Syst. Appl.*, vol. 68, pp. 36–44, 2017, doi: 10.1016/j.eswa.2016.10.009.

[20] U. Heute and A. S. Guzmán, “Removing ‘Cleaned’ eye-blinking artifacts from EEG measurements,” *2014 Int. Conf. Signal Process. Integr. Networks, SPIN 2014*, pp. 576–580, 2014.

[21] X. Jiang, G. Bin Bian, and Z. Tian, “Removal of artifacts from EEG signals: A review,” *Sensors (Switzerland)*, vol. 19, no. 5, pp. 1–18, 2019, doi: 10.3390/s19050987.

[22] L. Pion-Tonachini, S. H. Hsu, C. Y. Chang, T. P. Jung, and S. Makeig, “Online Automatic Artifact Rejection using the Real-time EEG Source-mapping Toolbox (REST),” *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. EMBS*, vol. 2018-July, pp. 106–109, 2018, doi: 10.1109/EMBC.2018.8512191.

[23] P. Li, Z. Chen, and Y. Hu, “A method for automatic removal of EOG artifacts from EEG based on ICA-EMD,” *Proc. - 2017 Chinese Autom. Congr. CAC 2017*, vol. 2017-Janua, pp. 1860–1863, 2017, doi: 10.1109/CAC.2017.8243071.

[24] E. Baehr, J. Peter Rosenfeld, R. Baehr, and C. Earnest, “Comparison of two EEG asymmetry indices in depressed patients vs. normal controls,” *Int. J. Psychophysiol.*, vol. 31, no. 1, pp. 89–92, 1998, doi: 10.1016/S0167-8760(98)00041-5.

Guiding vector fields in *Paparazzi* autopilot

Hector Garcia de Marina,[*] Murat Bronz, and Gautier Hattenberger[†]
Universidad Complutense de Madrid, Madrid, Spain
École National de l'Aviation Civile, Toulouse, France

ABSTRACT

This article is a technical report on the two different guidance systems based on vector fields that can be found in *Paparazzi*, a free *sw/hw* autopilot. Guiding vector fields allow autonomous vehicles to track paths described by the user mathematically. In particular, we allow two descriptions of the path with an implicit or a parametric function. Each description is associated with its corresponding guiding vector field algorithm. The implementations of the two algorithms are light enough to be run in a modern microcontroller. We will cover the basic theory on how they work, how a user can implement its own paths in *Paparazzi*, how to exploit them to coordinate multiple vehicles, and we finish with some experimental results. Although the presented implementation is focused on fixed-wing aircraft, the guidance is also applicable to other kinds of aerial vehicles such as rotorcraft.

1 INTRODUCTION

Autonomous aerial vehicles are presented as a great assistance to humans in challenging tasks such as environmental monitoring, search & rescue, surveillance, and inspection missions [1]. As mobile vehicles, they are *typically* commanded to travel from *point A* to *point B*. Even more, the requirements of the task at hand might demand a more precise route or *path* to be tracked while traveling from *A* to *B*. Guiding vector fields allow autonomous vehicles to track desired paths accurately with any temporal restriction. For example, we only assign the vehicle to visit a collection of connected points in the space (a geometric object); thus, we do not concern ourselves when the vehicle visits a specific point of such a geometric object. Guiding vector fields have been widely studied and employed in many different kinds of vehicles [2, 3, 4, 5, 6, 7].

Two guiding vector fields have been implemented in *Paparazzi*, an open-source drone hardware and software project encompassing autopilot systems and ground station software for multicopters/multirotors, fixed-wing, helicopters and hybrid aircraft that was founded in 2003 [8].

The first guiding vector field, or simply *GVF*, allows fixed-wing aircraft to track 2D (constant altitude) paths described by an implicit equation in *Paparazzi* [9]. This *GVF* guidance system has been exploited to coordinate a fleet of aircraft on circular paths [10]. The second guidance system is the parametric guiding vector field, or simply *p-GVF* [11]. This evolved version allows fixed-wing aircraft to track 3D paths described by a parametric equation in *Paparazzi*, and it also allows the coordination of multiple vehicles [12]. The main feature of the *p-GVF* is that it allows for tracking paths that are self-intersected, such an *eight figure*, and guarantees global convergence to the path. This *p-GVF* guidance systems has been exploited to characterize soaring planes.

Both implementations compensate for the disturbance of the wind on the vehicle by crabbing. Crabbing happens when the inertial velocity makes an angle with the nose heading due to wind. *Slipping* occurs when the aerodynamic velocity vector makes an angle (sideslip) with the body ZX plane. Slipping is (almost) always undesirable because it degrades aerodynamic performance. Crabbing is not an issue for the aircraft.

We split this paper into two equal parts focused on each guidance system. We will briefly show how the *GVF* and the *p-GVF* work and how they are implemented in *Paparazzi* so that a final user can define and try his own trajectories for fixed-wing aircraft. We present some performance results from actual telemetry, and we end with demonstrations concerning the coordination of more than one aircraft.

2 THE *GVF* GUIDANCE SYSTEM

2.1 *Theory*

This guidance system is based on constructing vectors tangential to the different level sets of the path in implicit form. Then, we add a normal component facing towards the direction of the desired level set. This will make a guiding vector that will drive the vehicle smoothly to travel on the desired path. Since we only care about the direction to follow and not the speed, we normalize the result to track a unit vector. This rationale is explained in Figure 1. We can express this technique formally as follows

$$\dot{p}_d(p) := \tau(p) - k_e e(p) n(p), \tag{1}$$

where $\frac{\dot{p}_d}{||\dot{p}_d||} \in \mathbb{R}^2$ is the unit vector to follow, $e \in \mathbb{R}$ is the current level set, $\tau \in \mathbb{R}^2$ and $n \in \mathbb{R}^2$ are the tangent and normal vectors respectively to the current level set, and $k_e > 0$ is a positive gain that defines how aggresive is the convergence of the guiding vector to the desired path. Note that all the variables in (1) depend on the position $p \in \mathbb{R}^2$ of the aircraft.

[*]Email addresseses: hgarciad@ucm.es
[†]Email addresses: {murat.bronz, gautier.hattenberger}@enac.fr

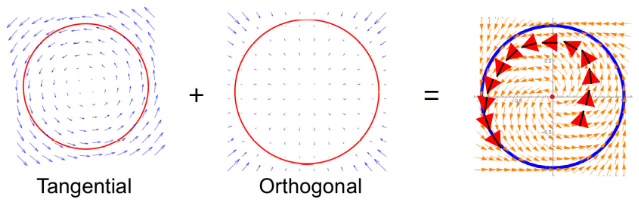


Figure 1: The *GVF* combines the vectors tangential and orthogonal to the level set of the desired path. The orthogonal part always points towards the desired path if the gradient of the level set is multiplied by the error quantity (*current level set - desired level set*).

In order to align the vehicle's velocity with the vector field in (1), the control action in *Paparazzi* will need of the gradient and the Hessian of the desired path [9]. In *Paparazzi* there is another gain k_n for a proportional controller to align both, the current velocity and the desired velocity.

2.2 *Paths for* GVF *in Paparazzi*

Let us illustrate this section with an example, and we will use it to demonstrate how the user has to write code to implement an arbitrary path in *Paparazzi*. Let us focus on a circle whose implicit equation is given by

$$\mathcal{P}(x,y) := x^2 + y^2 - r^2 = 0, \tag{2}$$

where r is the radius of the circle, and x and y the standard Cartesian coordinates. We first note that we define the desired level set as the zero level set. Therefore, the variable e in (1) can be identified as $e = \mathcal{P}(x,y)$. The gradient or $n(p)$ in (1) is trivially calculated as $n(p) = \begin{bmatrix}2x & 2y\end{bmatrix}$, and the Hessian $H(p) = \begin{bmatrix}2 & 0\\0 & 2\end{bmatrix}$.

All these path-dependent quantities must be codified in a file called *gvf/trajectories/gvf_circle.c*[1].

```c
void gvf_circle_info(float *phi, struct gvf_grad *
    grad, struct gvf_Hess *hess)
{

  struct EnuCoor_f *p = stateGetPositionEnu_f();
  float px = p->x;
  float py = p->y;

  // Parameters of the trajectory, circle's center
     and radius
  float wx = gvf_trajectory.p[0];
  float wy = gvf_trajectory.p[1];
  float r = gvf_trajectory.p[2];

  // Phi(x,y) or signal e
  *phi = (px-wx)*(px-wx) + (py-wy)*(py-wy) - r*r;

  // grad Phi
  grad->nx = 2 * xel;
  grad->ny = 2 * yel;

  // Hessian Phi
  hess->H11 = 2;
  hess->H12 = 0;
  hess->H21 = 0;
  hess->H22 = 2;
}
```

[1]The code can be found at https://github.com/paparazzi/paparazzi/tree/master/sw/airborne/modules/guidance

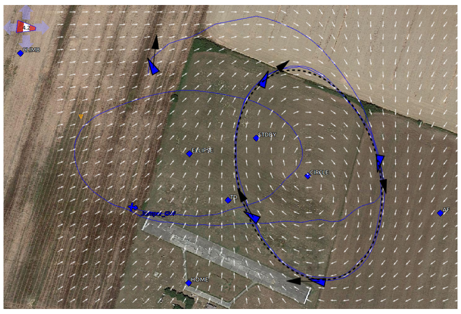


Figure 2: The *Paparazzi* ground control station showing different times for the position of an aircraft tracking a 2D ellipse with the *GVF* guidance system. The white arrows construct the vector field with unitary vectors calculated from (1).

Then, the user needs to define a high-level function in *gvf/gvf.c* to be called from the flight plan as follows

```c
bool gvf_circle_XY(float x, float y, float r)
{
  float e;
  struct gvf_grad grad_circle;
  struct gvf_Hess Hess_circle;

  gvf_trajectory.type = 1; // It is a circle
  gvf_trajectory.p[0] = x;
  gvf_trajectory.p[1] = y;
  gvf_trajectory.p[2] = r;

  gvf_circle_info(&e, &grad_circle, &Hess_circle);
  gvf_control.ke = gvf_circle_par.ke;
  gvf_control_2D(gvf_circle_par.ke, gvf_circle_par
    .kn, e, &grad_circle, &Hess_circle);

  gvf_control.error = e; // For telemetry

  return true;
}
```

The function *gvf_control_2D* calculates the desired roll angle to be tracked by the aircraft in order to align its velocity to (1). With only the definition of these two functions, together with the corresponding definitions in headers for the gains and used structs, is how a new path for the *GVF* guidance system is defined in *Paparazzi*.

2.3 *Performance in Paparazzi*

The figure 3 shows the described trajectory of an aircraft tracking a 2D ellipse in a windy environment. The airspeed of the aircraft was around 11m/s, and the windspeed was around 5 m/s.

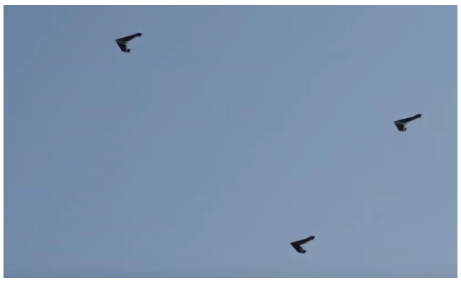


Figure 3: The *GVF* in *Paparazzi* can be employed to synchronize multiple aircraft in a distributed way. This caption corresponds to the 2017 IMAV competition.

2.4 Multi-vehicles

When the desired path is closed, such as a circle, then we can exploit the convergence properties of the *GVF* to synchronize different aircraft on the path. The different aircraft have to follow a positive or negative level set with respect to the desired path. While following a negative/positive level set, then the vehicle travels inside/outside the desired path and travels a shorter/larger distance in one lap. In that way, the aircraft can catch up or separate from each other. This can be done in a distributed way without any intervention from the Ground Control station. This implementation in *Paparazzi* has been discussed in detail in [13] and the theory can be found in [10].

3 The *p-GVF* guidance system

3.1 Theory

The guidance system *GVF* has a big inconvenience; the vector field (1) is not defined when the gradient is zero. We call this situation a singularity. For example, if $x = y = 0$ for the circle. That makes it impossible to implement paths with self-intersections. To avoid this difficulty, we have developed the *parametric Guiding Vector Field* or *p-GVF* [11]. We start from the parametric equation of the desired path, for example, for the circle

$$\begin{cases} x &= r\cos w \\ y &= r\sin w \end{cases}, \tag{3}$$

where $w \in \mathbb{R}$ is a free parameter. Then, we consider a $(2+1)$ dimensional path (two physical dimensions, and one virtual for w) as on the left side in Figure 4. Now, this new higher dimensional path can be seen in its implicit form for each coordinate so that we can apply a similar technique as in (1) again. In particular, the new implicit form for the circle is

$$e_x = x - r\cos w, \quad e_y = y - r\sin w, \tag{4}$$

and the vector field to be followed has the form

$$\xi = \nabla_\times e - \sum_{i=\{x,y\}} k_i e_i \nabla e_i, \tag{5}$$

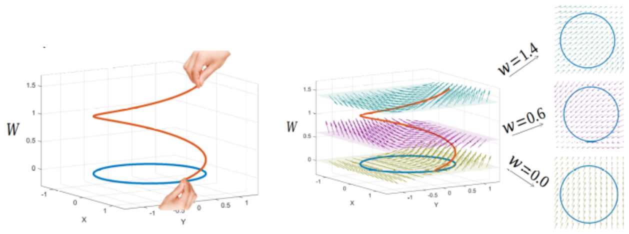


Figure 4: The p-GVF solves the singularity problem by taking a parametric description of the desired path. Then, the parameter w becomes a virtual dimension (topological surgery on the left image) so that we construct a singularity-free vector field following the tangential+orthogonal vector approach. The vehicle only needs to follow the projection of the vector field on the *physical world* coordinates.

where the first term to be explained shortly makes the vehicle to follow the path (*tangential component*), and the second one makes the vehicle to approach the path (*normal component*). The variable $e = \begin{bmatrix} e_x & e_y \end{bmatrix}^T$, ∇ is the gradient operator (note that $\nabla e_i = \begin{bmatrix} 0, \dots, 1, \dots, \frac{\partial e_i}{\partial w} \end{bmatrix}$), and

$$\nabla_\times e = (-1)^n \begin{bmatrix} \frac{\partial e_1}{\partial w} & \frac{\partial e_2}{\partial w} & \dots & \frac{\partial w}{\partial w} \end{bmatrix}^T. \tag{6}$$

Note that the last term $\frac{\partial w}{\partial w}$ sets the eventual velocity for w to one. Eventually, in *Paparazzi*, the desired velocity for w adapts to the actual speed of the vehicle. The main difference with respect to *GVF* is that the resultant guiding vector is not only driving the Cartesian coordinates but the virtual coordinate w. This can be seen on the right-hand side in Figure 4. We are free of singularities with this technique.

3.2 Paths for p-GVF in *Paparazzi*

Similarly, a user will need to define two main functions for an arbitrary path. The first one defines the parametric/implicit equations of the trajectory, i.e., e_x, e_y, and e_z, and its partial derivatives with respect to w for a 3D path. In the following example, we take a tilted circle where z_h and z_l define the maximum and minimum altitude of the circle of radius r. This function would be placed at *gvf_parametric/trajectories/gvf_parametric_3d_ellipse.c*.

```
void gvf_parametric_3d_ellipse_info(float *f1,
    float *f2, float *f3, float *f1d, float *f2d,
    float *f3d, float *f1dd, float *f2dd, float *
    f3dd)
{
  float xo = gvf_parametric_trajectory.
    p_parametric[0];
  float yo = gvf_parametric_trajectory.
    p_parametric[1];
  float r = gvf_parametric_trajectory.p_parametric
    [2];
  float zl = gvf_parametric_trajectory.
    p_parametric[3];
  float zh = gvf_parametric_trajectory.
    p_parametric[4];
  float alpha_rad = gvf_parametric_trajectory.
    p_parametric[5]*M_PI/180;

  float w = gvf_parametric_control.w;
  float wb = w * gvf_parametric_control.beta *
    gvf_parametric_control.s;

  // Parametric equations of the trajectory and
    the partial derivatives w.r.t. 'w'

  // These are e_x, e_y, and e_z
  *f1 = r * cosf(wb) + xo;
  *f2 = r * sinf(wb) + yo;
  *f3 = 0.5 * (zh + zl + (zl - zh) * sinf(
    alpha_rad - wb));

  // These are the partials of e_x, e_y, and e_z w
    .r.t. 'w'
  *f1d = -r * sinf(wb);
  *f2d = r * cosf(wb);
  *f3d = -0.5 * (zl - zh) * cosf(alpha_rad - wb);

  // These are the second partials of e_x, e_y,
    and e_z w.r.t. 'w'
  *f1dd = -r * cosf(wb);
  *f2dd = -r * sinf(wb);
  *f3dd = -0.5 * (zl - zh) * sinf(alpha_rad - wb);
}
```

Secondly, the following high-level function to be called from the flight plan must be placed at *guidance/gvf_parametric/gvf_parametric.cpp*.

```
bool gvf_parametric_3D_ellipse_XYZ(float xo, float
     yo, float r, float zl, float zh, float alpha)
{
  gvf_parametric_trajectory.type = ELLIPSE_3D;
  gvf_parametric_trajectory.p_parametric[0] = xo;
  gvf_parametric_trajectory.p_parametric[1] = yo;
  gvf_parametric_trajectory.p_parametric[2] = r;
  gvf_parametric_trajectory.p_parametric[3] = zl;
  gvf_parametric_trajectory.p_parametric[4] = zh;
  gvf_parametric_trajectory.p_parametric[5] =
    alpha;

  float f1, f2, f3, f1d, f2d, f3d, f1dd, f2dd,
    f3dd;

  gvf_parametric_3d_ellipse_info(&f1, &f2, &f3, &
    f1d, &f2d, &f3d, &f1dd, &f2dd, &f3dd);
  gvf_parametric_control_3D(
    gvf_parametric_3d_ellipse_par.kx,
    gvf_parametric_3d_ellipse_par.ky,
    gvf_parametric_3d_ellipse_par.kz, f1, f2, f3,
    f1d, f2d, f3d, f1dd, f2dd, f3dd);

  return true;
}
```

Note that we have a collection of positive gains to be tuned: k_x, k_y, and k_z, for the convergence of the different Cartesian coordinates.

3.3 Performance in Paparazzi

In figure 5 we show the performance of one aircraft tracking a simple Lissajous figure that is bent in 3D. The *p-GVF* controller passes two setpoints to low-level controllers, namely, the desired heading rate and the desired vertical speed. The first one is handled by controlling the roll angle of the aircraft depending on its actual ground speed. The second one is handled by controlling both the pitch and the throttle of the aircraft. Therefore, in order to have a good performance in *Paparazzi*, the vertical speed controller must be tuned in advance.

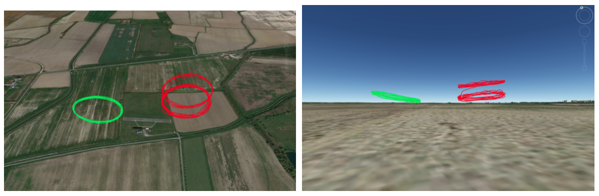


Figure 5: The *g-GVF* guidance system in *Paparazzi* allows aircraft to track 3D paths. These flights correspond to dynamic soaring experiments where the aircraft was tracking a tilted circle.

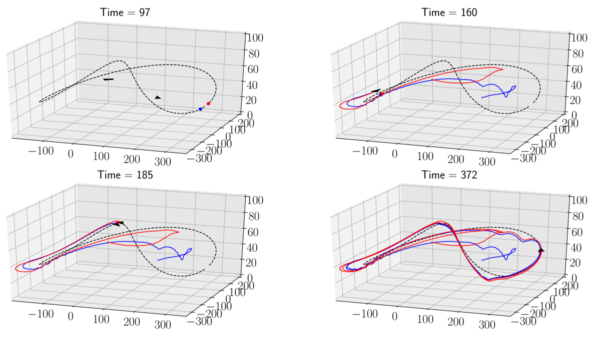


Figure 6: The *g-GVF* guidance system in *Paparazzi* allows aircraft to synchronize aircraft on 3D paths. In this example, both aircraft are instructed to have the same w so that they rendezvous.

3.4 Multi-vehicles

We can employ the *p-GVF* guidance system to synchronize vehicles on the desired path. Differently than with the *GVF*, now we will focus on controlling the *distances* between the virtual coordinates w for each vehicle. This is done by injecting the standard consensus algorithm to the control action, where each aircraft share their current w and compares the subtraction with the desired value [12]. The result makes the vehicles travel on the desired *parametric* path with their desired relative w between each other. This algorithm can run a distributed way, and there is no need for a Ground Control station in *Paparazzi*. In figure 6 we show the rendezvous of two aircraft on the same 3D path.

4 Conclusions and future work

This article presents a technical report on the two guiding systems based on guiding vector fields available in *Paparazzi* autopilot. We have shown how the user can implement its own desired path by mainly creating two functions in C code. The first function contains the mathematical information of the desired path, while the second function is what is called by the user from the Ground Control station. The presented guiding systems can be employed in *Paparazzi* to coordinate aircraft in a distributed way.

The future work focuses on extending the implementation in *Paparazzi* to other vehicles such as rotorcraft and rovers.

ACKNOWLEDGEMENTS

The work of Hector Garcia de Marina is supported by the grant *Atraccion de Talento* with reference number 2019-T2/TIC-13503 from the Government of the Autonomous Community of Madrid.

REFERENCES

[1] Guang-Zhong Yang, Jim Bellingham, Pierre E Dupont, Peer Fischer, Luciano Floridi, Robert Full, Neil Jacobstein, Vijay Kumar, Marcia McNutt, Robert Merrifield, et al. The grand challenges of science robotics. *Science robotics*, 3(14):eaar7650, 2018.

[2] V. M. Goncalves, L. C. A. Pimenta, C. A. Maia, B. C. O. Dutra, and G. A. S. Pereira. Vector fields for robot navigation along time-varying curves in n-dimensions. *IEEE Transactions on Robotics*, 26(4):647–659, Aug 2010.

[3] Adriano MC Rezende, Vinicius M Gonçalves, Guilherme V Raffo, and Luciano CA Pimenta. Robust fixed-wing UAV guidance with circulating artificial vector fields. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5892–5899. IEEE, 2018.

[4] Maciej Marcin Michałek and Tomasz Gawron. VFO path following control with guarantees of positionally constrained transients for unicycle-like robots with constrained control input. *Journal of Intelligent & Robotic Systems*, 89(1-2):191–210, 2018.

[5] Yuri A. Kapitanyuk, Sergey A. Chepinskiy, and Aleksandr A. Kapitonov. Geometric path following control of a rigid body based on the stabilization of sets. *IFAC Proceedings Volumes*, 47(3):7342 – 7347, 2014. 19th IFAC World Congress.

[6] Krzysztof Łakomy and Maciej Marcin Michałek. The VFO path-following kinematic controller for robotic vehicles moving in a 3D space. In *Robot Motion and Control (RoMoCo), 2017 11th International Workshop on*, pages 263–268. IEEE, 2017.

[7] Maciej Michalek and Krzysztof Kozlowski. Vector-field-orientation feedback control method for a differentially driven vehicle. *IEEE Transactions on Control Systems Technology*, 18(1):45–65, 2010.

[8] Gautier Hattenberger, Murat Bronz, and Michel Gorraz. Using the paparazzi uav system for scientific research. 2014.

[9] Hector Garcia De Marina, Yuri A Kapitanyuk, Murat Bronz, Gautier Hattenberger, and Ming Cao. Guidance algorithm for smooth trajectory tracking of a fixed wing UAV flying in wind flows. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5740–5745. IEEE, 2017.

[10] Hector Garcia De Marina, Zhiyong Sun, Murat Bronz, and Gautier Hattenberger. Circular formation control of fixed-wing uavs with constant speeds. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5298–5303. IEEE, 2017.

[11] Weijia Yao, Héctor Garcia de Marina, Bohuan Lin, and Ming Cao. Singularity-free guiding vector field for robot navigation. *IEEE Transactions on Robotics*, 2021.

[12] Weijia Yao, Hector Garcia de Marina, Zhiyong Sun, and Ming Cao. Distributed coordinated path following using guiding vector fields. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[13] Hector Garcia de Marina and Gautier Hattenberger. Distributed circular formation flight of fixed-wing aircraft with paparazzi autopilot. *arXiv preprint arXiv:1709.01110*, 2017.

Immersion and Invariance Based Trajectory Tracking Control of an Aerial Manipulation System

Aaron Lopez Luna[1]*, Hugo Rodríguez Cortés[2], Israel Cruz Vega[1], Jose Martinez-Carranza[1,3]
[1]Instituto Nacional de Astrofísica, Óptica y Electrónica, Puebla, Mexico. [2]Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, [3]Honorary Senior Research Fellow at the University of Bristol, Bristol BS8 1UB, UK

ABSTRACT

For aerial manipulators, steady flight must be guaranteed to perform safety interaction with the surrounding environment. This paper focuses on the development of a position control algorithm for an aerial manipulator system (AMS). The position control algorithm is based on the Immersion and Invariance (I&I) theory. The proposed controller maintains the position of the aerial manipulator at the desired point under external an internal disturbances. The control architecture uses the Visual-SLAM technique implemented using on-board sensors for AMS positioning. A series of outdoor experimental tests are performed to demonstrate the effectiveness of the proposed control strategy.

1 INTRODUCTION

The number of applications in which aerial robots are used has grown rapidly over the recent years. Aerial manipulators are designed to perform physical interaction with the surrounding environment. Aerial manipulation research focus on improving mechanical designs, and control strategies to increase the capability to manipulate objects, exert force on a surface and use tools in realistic applications [1]. However, technological challenges still need to be addressed before reliable use of such technology becomes possible. Among these challenges, the capability to carry on a manipulator, and handle disturbances during the movement of the manipulator is still a difficult challenge for small-size UAVs but represents an interesting research topic, and is the driving goal of this work.

The quadrotor is considered one of the most efficient UAV system for researchers around the world in the aerial manipulation field [2]. Due to the higher maneuvering capabilities of the quadrotor in comparison with other UAVs. A small-size quadrotor is capable of loading a light-weight manipulator and maintain a stable flight with a proper control algorithm [3].

The quadrotor is a nonlinear, open-loop, unstable, and underactuated system and the incorporation of a moving robotic arm to the structure increases the disturbance to the system that cannot be easily eliminated by feedback controllers. In order to tackle such problems and to improve the system performance researchers have proposed a host of advanced control methods to guarantee a stable flight during the manipulation task, including impedance control [4, 5, 6], Backstepping control [7, 8], and Proportional-Integral-Derivative (PID) control [9, 10] to name a few.

A methodology to design direct and indirect adaptive controllers for nonlinear systems, called Immersion and Invariance (I&I), was proposed in [11]. The I&I method is a control tool based on two classical theories, which are system Immersion and manifold Invariance [12]. The I&I approach captures the desired behavior of the system to be controlled by introducing a target dynamical system. Then, a suitable stabilizing control law is designed to guarantee that the controlled system asymptotically behaves like the target system [13]. Adaptive controllers based on I&I technique for UAV systems were developed in [14, 15] as a control solution to maintain the vehicle stable along the desired trajectory, and in [16] a multi-variable finite-time composite control strategy based on I&I was proposed for a quadrotor under unknown disturbances.

Up to now, the I&I control approaches for aerial systems described in the literature consider the problem of the position stability for the UAV but do not include the application of the I&I approach to improving the performance of an aerial manipulation system during the movement of the manipulator or the interaction phase, and considering for the aerial manipulator from the perspective of proposing one controller to solve the problem. In this paper, the divide and conquer strategy is followed. Thanks to the engineering advances in quadrotor control, there are available in the market aerial vehicles whose position dynamic response to operator inputs can be modeled by first or second order systems. Hence, in this paper, the internal commercial quadrotor controller is updated with an external I&I adaptive control-loop to solve the position control of an aerial manipulator.

The rest of this paper is organized as follows: The previous works of this research which present the development of the aerial manipulator and the Visual SLAM method to operate in outdoor scenarios are presented in section 2. In section 3 we describe the position control strategy based on the I&I theory to maintain the system in the desired position. The experimental set-up and the results of this work are presented in section 4. Finally, the conclusions of the results and a de-

*Email address(es): aaron.eleazar@inaoep.mx scription of the future work of this research are presented in section 5.

2 Background

Before beginning with the present research work, in this section, we mention a background of the investigation, experiments, and results in the aerial manipulation field we achieved in recent years. At the beginning of of the research, we developed an aerial manipulator based on a Commercial small UAV (parrot bebop-2) with a 2-DOF arm to study the behavior of the system with the added weight and the disturbances of the arm. The manipulator was designed and implemented based on the technical capabilities of the selected aerial platform. Then, when a stable flight was achieved through a computational compensation control, we improve the capabilities of the proposed aerial manipulator implementing an external control-loop based on a Gain-Scheduling (GS) PID control strategy to work with the internal bebop-2 controller and mitigate the disturbances induced by the movement of the 2-DOF arm and the contact with a vertical surface.

Rojas-Perez and Martinez-Carranza presented in [17] an obstacle avoidance system based on the Visual SLAM approach to estimate the position of a bebop-2 taking advantage of the on-board camera, and without any other motion capture system to feedback the position of the vehicle. We combined this estimation pose technique with our aerial manipulation system to extend its capabilities to outdoor scenarios where the use of caption motion systems to sensing the pose of the vehicle are limited. Then, we presented a study of the behavior of the system in interaction with a vertical surface in an outdoor environment. To improve the results of previous works we consider the implementation of the I&I as an alternative to the GS-PID external control due to the capability to increase the robustness of a nonlinear system against disturbances [11]. In the following section, we present the I&I methodology and the implementation of our aerial manipulation system.

3 The Immersion and Invariance Control Strategy

The use of the I&I approach for stabilization of nonlinear system was presented in [11]. Before explaining the implementation of this research, we briefly recall the fundamental conditions for the standard I&I controller design. Consider the following system:

$$\dot{x} = f(x) + g(x)u \tag{1}$$

with state $x \in \mathbb{R}^n$ and control $u \in \mathbb{R}^m$, with an equilibrium point $x_* \in \mathbb{R}^n$ to be stabilized. Let $p < n$ and assuming existence of mappings $\alpha(\cdot) : \mathbb{R}^p \to \mathbb{R}^p, \pi(\cdot) : \mathbb{R}^p \to \mathbb{R}^n, c(\cdot) : \mathbb{R}^p \to \mathbb{R}^m, \phi(\cdot) : \mathbb{R}^n \to \mathbb{R}^{n-p}, \psi(.,.) : \mathbb{R}^{nx(n-p)} \to \mathbb{R}^m$ such that:

The system (target system):

$$\dot{\xi} = \alpha(\xi) \tag{2}$$

With $\xi \in \mathbb{R}^p$ has a globally asymptotically stable equilibrium at $\xi_* \in \mathbb{R}^p$ and $x_* = \pi(\xi_*)$. (Immersion condition) For all $\xi \in \mathbb{R}^p$

$$f(\pi(\xi)) + g(\pi(\xi))c(\xi) = \frac{\partial \pi}{\partial \xi}\alpha(\xi) \tag{3}$$

(Implicit manifold) The following set identity holds:

$$x \in \mathbb{R}^n \mid \phi(x) = 0 = x \in \mathbb{R}^n \mid x = \pi(\xi), \xi \in \mathbb{R}^p \tag{4}$$

(Manifold attractivity and trajectory boundedness) The system:

$$\dot{z} = \frac{\partial \phi}{\partial x}(f(x) + g(x)\psi(x,z)) \tag{5}$$

With state z, has a globally asymptotically stable equilibrium at zero uniforly in x. Further, the trajectories of the system

$$\dot{x} = f(x) + g(x)\psi(x,z) \tag{6}$$

are bounded for all $t \in [0.\infty)$ Then, x_* is a globally asymptotically stable equilibrium of the closed loop system $\dot{x} = f(x) + g(x)\psi(x, \phi(x))$.

The result summarized above lends itself to the following interpretation. Given the system 1 and the target dynamical system 2 find, if possible, a manifold $\mathcal{M}$, described implicitly by $\{x \in \mathbb{R}^n \mid \phi(x) = 0\}$, and in parameterized form by $\{x \in \mathbb{R}^n \mid x = \pi(\xi), \xi \in \mathbb{R}^p\}$, which can be rendered invariant and asymptotically stable, and such that the (well defined) restriction of the closed loop system to $\mathcal{M}$ is described by $\dot{\xi} = \alpha(\xi)$. Notice, however, that we do not propose to apply the control $u = c(\xi)$ that renders the manifold invariant, instead we design a control law $u = \psi(x, z)$ that drives to zero the off-the-manifold coordinate z and keeps the system trajectories bounded. (For the methodology proof, see [11]).

3.1 *Application to an Aerial Manipulation system*

Aerial vehicle's internal autopilots shape the system's response at different levels depending on the available sensors. Hence, if rotational states are measured, the internal autopilot can shape the aerial vehicle's response as a second-order system. If, in addition, the translational states are measured, the aerial vehicle can commanded to behave as a first-order system. This is the case of the aerial vehicle used in this work. Using the information from the Attitude Reference and Hedging System as well as optical flow and SLAM algorithms, the internal controller shapes the aerial vehicle as a first-order dynamics expressed by

$$\frac{dx}{dt} = u_\theta + \delta \tag{7}$$

We can assume that our aerial manipulation system can be represented by equation 7. The experimental validation of this model is presented in the following section. δ represents the disturbance of the robotic arm, the wall effect, the interaction with a surface, or any other disturbance. Now, we describe the steps to implement the I&I control strategy to the proposed aerial manipulation system. We first defined the estimation error as

$$\tilde{\delta} = \delta - \rho + \beta(x) \tag{8}$$

where $\tilde{\delta}$ is the estimation error. The time derivative of the estimation error gives

$$\frac{d\tilde{\delta}}{dt} = -\frac{d\rho}{dt} + \frac{\partial\beta}{\partial x}\left(u_\theta + \delta\right) \tag{9}$$

Replacing δ from

$$\frac{d\tilde{\delta}}{dt} = -\frac{d\rho}{dt} + \frac{\partial\beta}{\partial x}\left(u_\theta + \tilde{\delta} + \rho - \beta\right) \tag{10}$$

Now, the dynamics of the estimator' state is defined in terms of known signals as follows

$$\frac{d\rho}{dt} = \frac{\partial\beta}{\partial x}\left(u_\theta + \rho - \beta\right) \tag{11}$$

Substituting (11) in (10) one gets

$$\frac{d\tilde{\delta}}{dt} = \frac{\partial\beta}{\partial x}\tilde{\delta} \tag{12}$$

To ensure that the estimator error converges to zero, one selects $\beta(x) = \Gamma x$, with Γ a positive constant, thus,

$$\frac{d\tilde{\delta}}{dt} = -\Gamma\tilde{\delta} \tag{13}$$

To compensate the disturbance, the following controller is proposed

$$u_\theta = -k_p(x_d - x) - \rho + \beta(x) \tag{14}$$

where $\tilde{x} = x - x_d$ with x_d the desired constant reference, this is,

$$u_\theta = -k_p(x_d - x) - \rho - \Gamma x \tag{15}$$

moreover,

$$\frac{d}{dt}\rho = -\Gamma(\rho + \Gamma x) \tag{16}$$

The closed-loop dynamics reads as

$$\begin{aligned} \frac{d\tilde{x}}{dt} &= -k_p\tilde{x} + \tilde{\delta} \\ \frac{d\tilde{\delta}}{dt} &= -\Gamma\tilde{\delta} \end{aligned} \tag{17}$$

Hence, for any positive gains k_p, Γ, the error signals $\tilde{x}$ and $\tilde{\rho}$ converge to zero. Thanks to the quadrotor symmetry, the controller for the y position is designed following exactly the same procedure. The control law expressed by equations (14)-(16) are implemented in the aerial manipulator to control the longitudinal and lateral motions of the system to regulate the position in an outdoor scenario and guarantee a stable flight. In the next section, we describe the setup of the experiments to investigate the effectiveness of the proposed control strategy.

4 Experiments and Results

4.1 System Behavior

To validate the assumption that the proposed aerial manipulator can be modeled as a first or second order system, a set of tests were developed to investigate the response of the system to an input signal. First, the bebop-2 was tested to prove that the internal controller works efficiently. Then, we repeated the experiment with the same signal but now with the robotic arm attached to bebop-2 to compare the influence of the robotic arm in the system. Figure 1 shows the sequence of movements the system performs in each experiment when the input signal is sent to the vehicle.

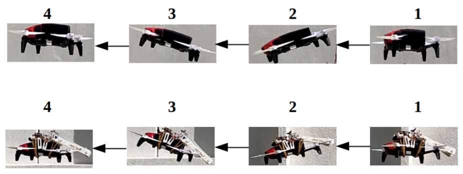


Figure 1: Sequence of movements. Response of the system to input signal.

Several tests were carried out to obtain a mean result of the behavior of the system in both experiments. Figure 2 shows the most representative results in the first experiment.

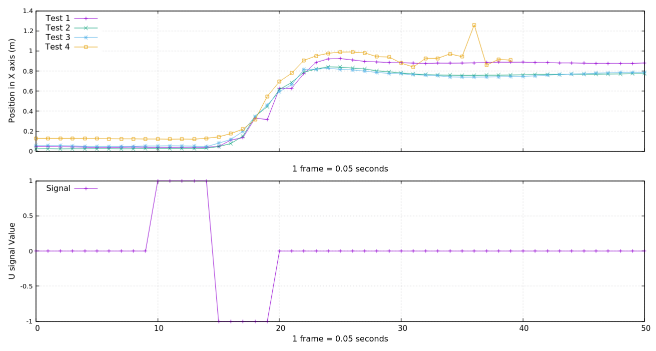


Figure 2: Behavior of bebop-2. Response to input signal.

The results of the second experiment are shown in Figure 3, as in the first experiment a set of tests were carried out. The most representative results of the behavior of the aerial manipulator to the test signal are presented in the graphic. According to the results shown in the graphics, we proved the system behaves as a firs order system even with the robotic arm attached to the structure. therefore, we can assume our proposed aerial manipulator can be described by equation 7

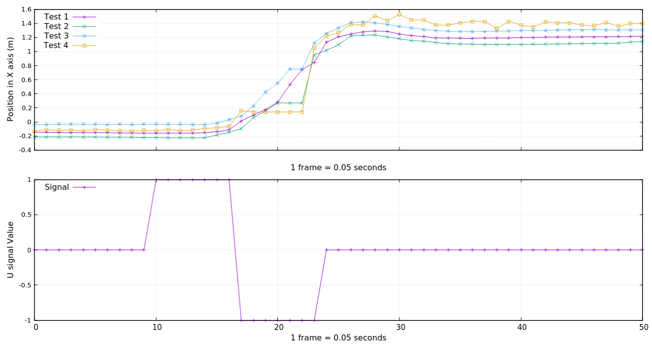


Figure 3: Behavior of the aerial manipulator. Response to input signal.

4.2 *Experimental Setup*

The proposed control strategy was proved via experimental tests in an outdoor environment. To reduce the disturbances (mostly the disturbances due to the environment) an I&I controller was implemented and added to the system. In previous works, we used the VICON camera system to measure the position of the vehicle and feedback the information to the controller [3, 18] but, in outdoor environments, the implementation of VICON is laborious and limited. For that reason, in most recent works we incorporated the Visual SLAM technique to replace the VICON system in the control structure and to improve the capabilities of our proposed system [19]. The pose estimation using visual SLAM was exploited in [17, 20]. The method takes advantage of two characteristics of the aerial vehicle; the on-board camera, and the altitude measurement.

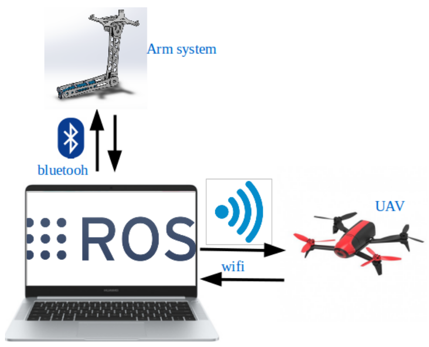


Figure 4: Communication of the complete system.

Due to the features of the aerial platform, we can include the Visual SLAM technique as a pose estimation method in the complete system. Figure 4 shown the communication scheme. The control algorithm runs in a ground station (computer), and we take advantage of the wifi and Bluetooth connection to communicate the computer with the robotic arm and the bebop-2. The coordinate frame of the aerial manipulator is represented by Figure 5. The longitudinal and lateral motions of the system are represented by the X and Z axis. The I&I controller was implemented for both axes, a standard PID controller was also incorporated to control θ and Y axis and guarantee the correct orientation and distance to the ground.

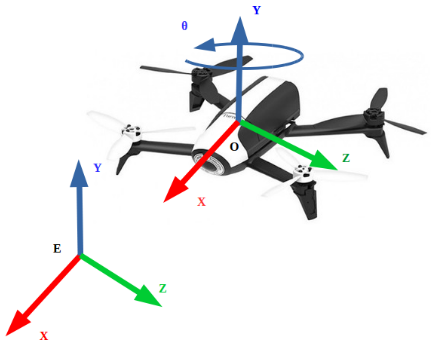


Figure 5: Aerial manipulator scheme.

The outdoor scenario can be shown in Figure 6. In this research work, we decided to put the robotic arm on the top of the vehicle, this allows the sensors below bebop-2 to operate without any kind of obstruction and guarantee a good measurement. This change in the position of the robotic arm also reduces the load because the extension legs are not more needed. The carpet below the aerial manipulator provides visual features which ensure a good performance of the pose estimation method and reduce the pose estimation error.

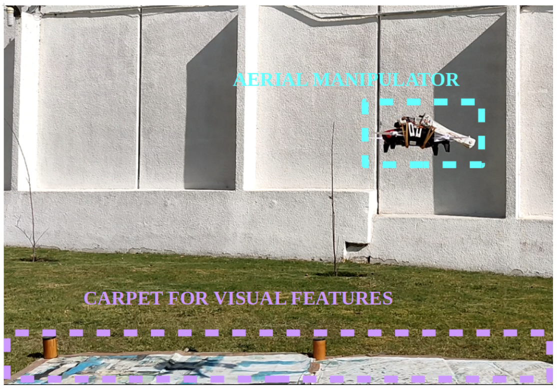


Figure 6: Outdoor scenario. A carpet is set in the ground to provide more texture for the SLAM technique and improve the pose estimation.

Figure 7 depicts the block diagram for this research. The pose of the system when is in fly mode is provided for the Visual SLAM technique. The movement of the robotic arm to fold and unfold produces a set of different disturbances. The objective of the attitude control is to reduce the displacement produced by this disturbance and any other disturbance induced by the outside conditions to maintain the aerial vehicle in the desired position. The proposed controller operates as external control, calculating the control signal U for the X and Z axis to displace the vehicle to the desired position, then, the U signals are sent to the internal control of Bebop-2. Finally, the pose estimation provides by the Visual SLAM closes the external control loop.

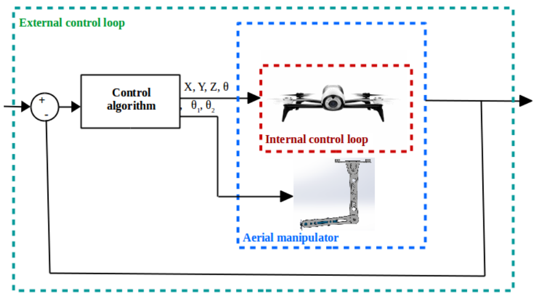


Figure 7: Block diagram of the proposed system.

4.3 Results

Through several experimental tests, we studied the behavior of the system and tuned the parameters k_p and Γ of the control law expressed by equations (14)-(16) to improve the performance of the complete system. In Figure 8 it can be shown the response of the system in the first experiments when the value of the parameters was chosen according to our experience.

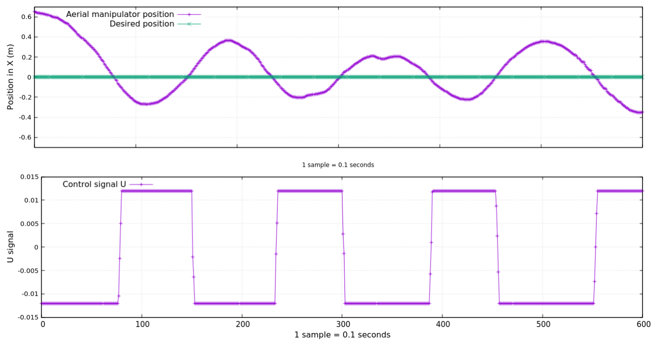


Figure 8: Behavior of the aerial manipulator with the I&I control strategy. First results.

Due to the facility provided by the I&I technique, after a few experiments, we were able to tune the value of the parameters and ensure the system reaches the desired position, and maintain the pose during the disturbances induced by the environment and the robotic arm. Figure 9 shown the results of the final experiments.

5 Conclusions and Future Work

In this work, a control strategy based on immersion and invariance technique has been designed and implemented in order to control an aerial manipulator to tackle the issues related to the environment and the movement of the robotic arm. The experimental results obtained for several tests are quite promising. A merit of the proposed algorithm is shown by the equations of the control law which reduced the number of parameters of the controller, simplifying the experimental phase and allowing to obtain an efficient performance of the system. According to the results obtained in this work, we consider that the control can be tested near to a vertical surface and aiming to make contact with the end effector of the arm, this could represent a novel approach for the aerial interaction field.

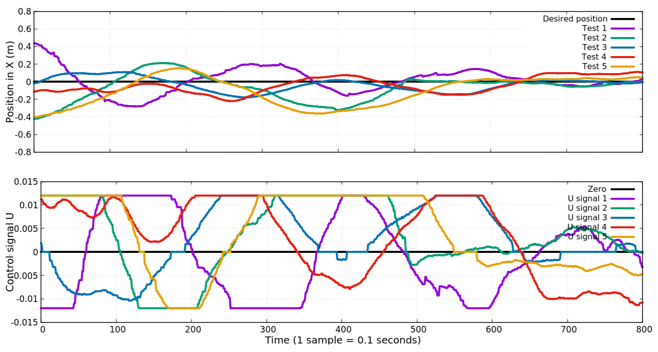


Figure 9: Behavior of the aerial manipulator with the I&I control strategy. Final results.

References

[1] T. Bartelds, A. Capra, S. Hamaza, S. Stramigioli, and M. Fumagalli. Compliant aerial manipulators: Toward a new generation of aerial robotic workers. *IEEE Robotics and Automation Letters*, 1(1):477–483, Jan 2016.

[2] Tiehua Wang, Kazuki Umemoto, Takahiro Endo, and Fumitoshi Matsuno. Dynamic hybrid position/force control for the quadrotor with a multi-degree-of-freedom manipulator. *Artificial Life and Robotics*, 24(3):378–389, Sep 2019.

[3] A. L. Luna, I. C. Vega, and J. Martinez-Carranza. Towards micro aerial manipulation using a computational compensation strategy. In *10th International Micro Air Vehicle Competition and Conference (IMAV 2018)*, 11 2018.

[4] F. Huber, K. Kondak, K. Krieger, D. Sommer, M. Schwarzbach, M. Laiacker, I. Kossyk, S. Parusel, S. Haddadin, and A. Albu-Schiffer. First analysis and experiments in aerial manipulation using fully actuated redundant robot arm. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3452–3457, Nov 2013.

[5] C. Korpela, P. Brahmbhatt, M. Orsag, and P. Oh. Towards the realization of mobile manipulating unmanned aerial vehicles (mm-uav): Peg-in-hole insertion tasks. In *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, pages 1–6, April 2013.

[6] H. Lee, S. Kim, and H. J. Kim. Control of an aerial manipulator using on-line parameter estimator for an unknown payload. In *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 316–321, Aug 2015.

[7] S. Di Lucia, G. D. Tipaldi, and W. Burgard. Attitude stabilization control of an aerial manipulator using a quaternion-based backstepping approach. In *Mobile Robots (ECMR), 2015 European Conference on*, pages 1–6, Sept 2015.

[8] Andrew Zulu and Samuel John. A review of control algorithms for autonomous quadrotors. *CoRR*, abs/1602.02622, 2016.

[9] Aaron Dollar Paul E. I. Pounds. Hovering sability of helicopters with elastics constraints. *ASME Dynamic Systems and Control Conference*, 2010.

[10] Matko Orsag, Christopher Korpela, and Paul Oh. Modeling and control of mm-uav: Mobile manipulating unmanned aerial vehicle. *Journal of Intelligent & Robotic Systems*, 69(1):227–240, 2013.

[11] A. Astolfi and R. Ortega. Immersion and invariance: a new tool for stabilization and adaptive control of nonlinear systems. *IEEE Transactions on Automatic Control*, 48(4):590–606, 2003.

[12] Jinchang Hu and Honghua Zhang. Immersion and invariance based command-filtered adaptive backstepping control of vtol vehicles. *Automatica*, 49:2160–2167, 07 2013.

[13] Lei Wang, Fulvio Forni, Romeo Ortega, and Hongye Su. Immersion and invariance stabilization of nonlinear systems: A horizontal contraction approach. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 3093–3097, 2015.

[14] Y. Bouzid, H. Siguerdidjane, and Y. Bestaoui. Real time autopilot based on immersion invariance for autonomous aerial vehicle. *IFAC-PapersOnLine*, 49(17):176–181, 2016. 20th IFAC Symposium on Automatic Control in AerospaceACA 2016.

[15] Gilles Tagne, Reine Talj, and Ali Charara. Immersion and invariance control for reference trajectory tracking of autonomous vehicles. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 2322–2328, 2013.

[16] Xinkai Li, Hongli Zhang, Wenhui Fan, Jiawei Zhao, and Cong Wang. Multivariable finite-time composite control strategy based on immersion and invariance for quadrotor under mismatched disturbances. *Aerospace Science and Technology*, 99:105763, 02 2020.

[17] L. O. Rojas-Perez and J. Martinez-Carranza. Metric monocular slam and colour segmentation for multiple obstacle avoidance in autonomous flight. In *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, pages 234–239, Oct 2017.

[18] A. L. Luna, I. C. Vega, and J. Martinez-Carranza. Gain-scheduling and pid control for an autonomous aerial vehicle with a robotic arm. In *2018 IEEE 2nd Colombian Conference on Robotics and Automation (CCRA)*, pages 1–6, Nov 2018.

[19] A. L. Luna, J. Martinez-Carranza, and I. C. Vega. Towards aerial interaction of mavs in gps-denied environments. In *2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS)*, pages 113–121, Nov 2019.

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

XINCA: Extended Incremental Non-linear Control Allocation on a Quadplane

H.J. Karssies, and C. De Wagter*
Delft University of Technology, Micro Air Vehicle Lab, Kluyverweg 1, 2629HS Delft, the Netherlands

ABSTRACT

Controlling over-actuated Unmanned Aerial Vehicle (UAV) is an important task to achieve reliable fail-safe autonomous flight. Incremental Non-linear Control Allocation or INCA has been proposed to solve the platform's control allocation problem by minimizing a set of objective functions with a method known as the Active Set Method. This work proposes an extension to INCA to control the outer loop of a quadplane UAV, an in-plane combination between a quadrotor and a conventional fixed-wing. The novel controller is called Extended INCA or XINCA and optimizes a mix of physical actuator commands and angular control setpoints fed to the vehicle's inner loop. It does so while adapting to varying flight phases, conditions, and vehicle states, and taking into account the aerodynamic properties of the main wing. XINCA has low dependence on accurate vehicle models and requires only several optimization parameters. Flight simulations and experimental flights are performed to prove the performance of both controllers.

1 INTRODUCTION

Unmanned Arial Vehicle or Unmanned Aerial Vehicle (UAV)s have gained a tremendous amount of popularity. Not only have they proven to be valuable research platforms and entertaining toys, they have also found many other applications in fields like defence [1], surveillance [2], medical assistance [3], transportation of both goods and humans [4], agriculture [5], inspection [6], mapping [7], and many others.

Some challenges that are often faced in UAV design are endurance, reliability, versatility, and affordability. Existing solutions often perform well on some but not all of these aspects. Fixed-wing aircraft like the ones by [8, 9] and [10] for instance master endurance as a result of the passive wing-induced lift that keeps them airborne. Rotorcraft on the other hand, like designs by [11, 12] and [13], are much more versatile since they can hover, take off and land vertically. They are also inexpensive to produce, mechanically simple and their control has been well solved. Their powered generation of lift however severely limits their endurance, and designs like the conventional quadcopter typically have multiple single points of failure. It is therefore that many researchers have come up with hybrid platforms, that aim to combine the best of different worlds.

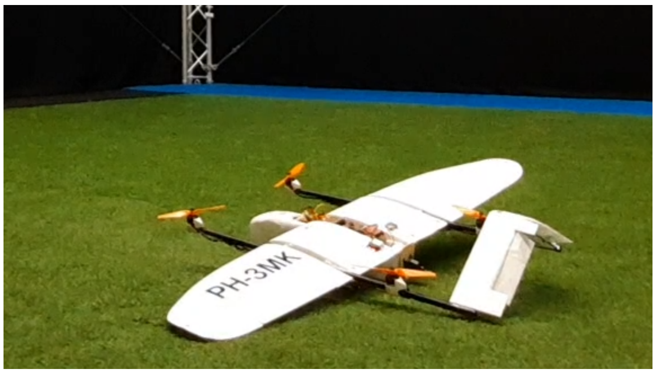


Figure 1: The TU Delft Quadplane in the Cyberzoo.

Some examples of hybrid platforms include tilt rotor/wing UAVs, tail sitters, transformable UAVs, and quadplanes. Tiltrotor/wing UAVs like designs by [14] and [15] mechanically change the orientation of their propulsion units to either generate lift during vertical take-off and landing or horizontal thrust while flying horizontally with wing induced lift. Similarly, tail sitters as discussed by [16] and [17] change the orientation of the entire vehicle when transitioning from vertical take-off and landing orientation to horizontal flight. This reduces the mechanical complexity of the system, resulting in a more reliable, lighter, and cheaper platform, albeit at the cost of sensitivity to wind gusts. A completely different class of hybrid UAVs are the ones that are transformable like the one designed by [18]. By changing the configuration of the entire vehicle, they can transform between very different types of UAVs, like for instance a monocopter and a fixed-wing aircraft.

Lastly, a common class of hybrid UAVs is formed by quadplanes, like the one used as an experimental platform for this research (See Figure 1). Earlier designs include those by [19, 20, 21, 22] and [23]. The quadplane has a static configuration with both upward-facing rotors for vertical take-off and landing and fixed wings with a horizontal propulsion unit for horizontal flight. Despite the added weight of flight phase-specific actuators, its mechanical simplicity makes this versatile and enduring vehicle a promising research platform.

*Email address(es): c.dewagter@tudelft.nl

Making such a Quadplane fly as efficiently and safely as possible poses a number of challenges. These include dealing with large flight envelopes, over-actuation, its non-linear nature, and its sensitivity to wind gusts. The quadplane used for this research and its control challenges are described in Section 2. An existing control method called Incremental Nonlinear Control Allocation (INCA) is discussed in Section 3, and its optimization methods in Section 4. A proposed extension of this control method, called Extended Incremental Non-linear Control Allocation (XINCA), is presented in Section 5. The implementation of the INCA and XINCA controllers on the TU Delft Quadplane is shown in Section 6, and Sections 7 and 8 respectively present results from simulations and test flights performed using this novel control method. Lastly, Section 9 gives the conclusions.

2 The TU Delft Quadplane

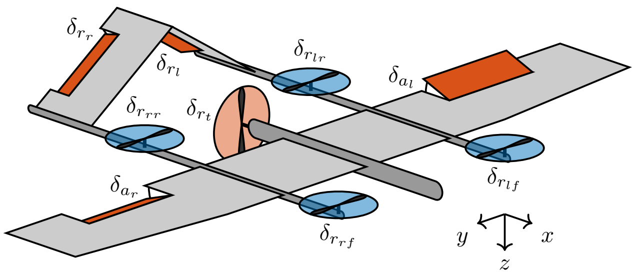


Figure 2: Overview of the nine quadplane actuators
🟦 = quadcopter actuator set, 🟧 = fixed wing actuator set

The quadplane is a hybrid of a fixed-wing aircraft and a quadcopter. A conventional example of a quadplane is the one used for this research, the *TU Delft Quadplane*. A schematic representation of this platform is shown in Figure 2. It shows the quadplane's nine actuators: four upward-facing rotors that could be considered as the *quadcopter actuator set*, and four control surfaces, and a tail rotor that could be considered the *fixed-wing actuator set*. Having actuator sets that can operate simultaneously, quadplanes are considered *over-actuated*. Literature shows that this over-actuation is often dealt with by using only one actuator set during specific flight phases, and only briefly combining them during a transition phase between vertical and horizontal flight [19, 20, 21, 22].

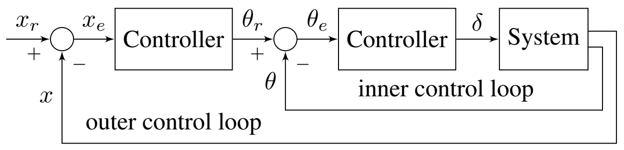


Figure 3: Simplified schematic UAV controller diagram (x = position, θ = attitude, δ = system input)

UAV controllers often use cascaded outer and inner loops shown in Figure 3. The outer loop, also called the position or guidance loop, controls the position error and outputs a reference attitude. The inner loop or attitude loop controls actual attitude and uses that to allocate control to suitable actuators. This allocation is quite straightforward when the vehicle is not over-actuated or when only a single actuator set is used.

Quadplanes could however fly more efficiently when continuously assessing each actuator's suitability to satisfy a certain control demand. This assessment should take into account each actuator's effectiveness based on the system's states, but could also penalize large deviations from preferred actuator positions. Such an optimization problem is known as a control allocation problem. The advantages are that first, it can minimize the control effort of a UAV, potentially resulting in more efficient flight and enhanced flight endurance. The other advantage is that when certain actuators are saturating, it can allocate control to other actuators to still satisfy a given control demand, resulting in safer and more reliable flight. The control allocation method used in this research is called Incremental Non-linear Control Allocation or INCA, which solves the inner loop's control allocation optimization problem and is presented in Section 3.

Another challenge in controlling quadplanes is caused by the fundamentally different outer loop dynamics of the quadplane during different flight phases. When flying as a quadcopter, for instance, a change in pitch angle causes the quadplane to accelerate in a longitudinal direction. When flying as a fixed-wing aircraft, however, a change in pitch will cause the quadplane to either climb or descent. Furthermore, the quadplane is over-actuated in its outer loop as well as its inner loop, since it can control a positive forward acceleration during hovering with both its pitch angle and pusher rotor. The latter is often preferable since negative pitching maneuvers might introduce an undesirable negative wing-induced lift. A positive backward acceleration however is only achievable by pitching the quadplane backward. To address the challenges named above, an extension of the INCA controller is presented in Section 5, which performs an outer loop optimization similar to the INCA inner loop optimization. This method is called Extended Incremental Non-linear Control Allocation, or XINCA.

3 INCA

Incremental Non-linear Control Allocation or INCA, is a very promising control allocation algorithm. It has already theoretically been demonstrated on over-actuated vehicles like the Lockheed Martin Innovative Control Effector aircraft by [24]. [25] have proven the control method to be effective in actual flight on non-over-actuated quadcopters. The architecture of INCA augments a method called Nonlinear Dynamic Inversion, or Non-linear Dynamic Inversion (NDI). NDI measures a vehicle's states and uses an accurate model to predict angular and linear accelerations as a result of these states. Their difference with the vehicle's desired accelerations is then used to calculate appropriate control inputs

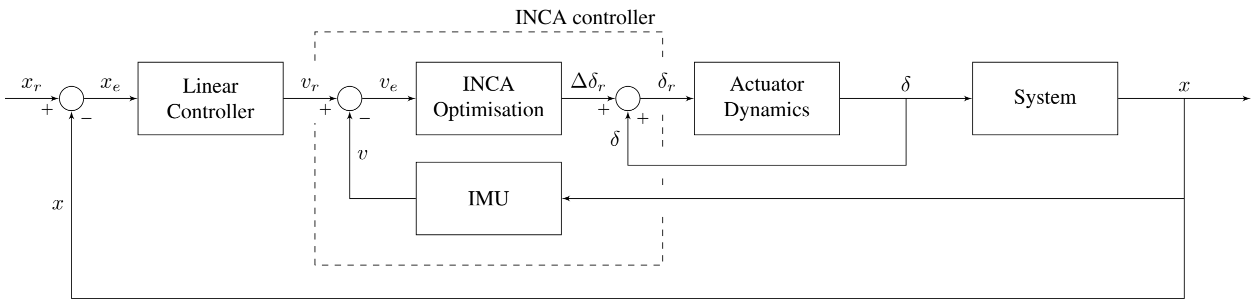


Figure 4: A schematic representation of an INCA controller (x = state vector, v = virtual input, δ = control input vector)

using reliable actuator models. A successful example of an implementation of NDI is the work by [26].

However effective, NDI highly relies on detailed and accurate models of the vehicle it controls. A variation on this approach provides a solution to this problem and is called Incremental Non-linear Dynamic Inversion, or Incremental Non-linear Dynamic Inversion (INDI) [25]. While it still relies on an actuator model, instead of using a vehicle model to predict its angular and linear accelerations as a result of its states, it uses inertial measurement data to observe these accelerations. And the control effectiveness model does not need to be as accurate, since the controller will compensate for any unexpected effects of the actuators by incrementing.

An example where INDI has been proven successfully in quadcopter flight is presented by [27].

Both NDI and INDI invert actuator effectiveness models in order to calculate appropriate actuator commands. When dealing with over-actuated UAVs however, it is mathematically challenging to derive appropriate actuator commands by simply inverting these actuator effectiveness models, since any calculated actuator command solution is no longer singular, and there exist infinite solutions. INCA deals with this by expressing this control allocation problem as an optimization problem, that needs to be solved by minimizing a certain cost function. A schematic representation of INCA is shown in Figure 4. Like an INDI controller, INCA uses the difference between desired accelerations and inertial measurements to determine an incremental control demand, also known as the virtual input to the INCA optimization. The optimization scheme then calculates an optimal actuator increment to satisfy the control demand. Note that while the rotor effectiveness is relatively constant, the effectiveness of the aerodynamic surfaces is proportional to the square of the true airspeed. The optimization method is presented in Section 4.

4 INCA Optimization

Let $\mathbf{H}$ be a matrix containing the linearized effectiveness of all actuators, and τ_c the control demand that will be used as virtual input to the INCA optimization. An unconstrained control command increment $\Delta\delta$ should then always satisfy the following equation:

$$\mathbf{H}\Delta\delta = \tau_c \tag{1}$$

When this increment is constrained by actuator limits, an error between the control demand and the achieved control might occur, but should still be minimized. Also minimizing control effort, i.e., the difference between actual actuator increments $\Delta\delta$ and preferred actuator increments $\Delta\delta_p$, yields:

$$\underset{\Delta\delta}{min} \ \|\gamma \mathbf{W}_\tau(\mathbf{H}\Delta\delta - \tau_c)\|_2 + \|\mathbf{W}_\delta(\Delta\delta_p - \Delta\delta)\|_2 \tag{2a}$$

$$\text{subject to } \Delta\delta_{min} \leq \Delta\delta \leq \Delta\delta_{max} \text{ and } \dot{\delta} \leq \dot{\delta}_{max} \tag{2b}$$

where $\mathbf{W}_\tau$ and $\mathbf{W}_\delta$ are weighing matrices to prioritize selected control demands and actuators over others, and γ is a constant that prioritizes one sub-objective over the other. This type of objective function is called a *Quadratic Program* and can include as many separate sub-objectives as needed. Quadratic Programming is often used for Control Allocation problems. [28] presents proof that it can provide automatic redistribution of control in case of actuator saturation. [24] and [27] both apply it, on a modern fighter jet and a quadcopter UAV respectively. The objective function is often rewritten to a standardized quadratic form, which many solvers can easily work with:

$$\underset{\Delta\delta}{min} \ \Delta\delta^T \mathbf{Q}\Delta\delta + c^T\Delta\delta \tag{3a}$$

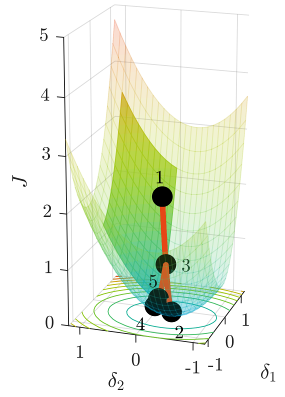
$$\text{subject to } \mathbf{A}\Delta\delta \leq b \tag{3b}$$

where $\mathbf{Q} = \mathbf{F}^T\mathbf{F}$, $c = 2\mathbf{F}^T g$,

$$\mathbf{F} = \begin{pmatrix} \gamma\mathbf{W}_\tau\mathbf{H} \\ \mathbf{W}_\delta \end{pmatrix}, \ g = \begin{pmatrix} \gamma\mathbf{W}_\tau\tau_c \\ \mathbf{W}_\delta\Delta\delta_p \end{pmatrix},$$

$$\mathbf{A} = \begin{pmatrix} \mathbf{I} \\ -\mathbf{I} \end{pmatrix} \text{ and}$$

$$b = \begin{pmatrix} \min(\delta_{max} - \delta_0, \dot{\delta}_{max}\Delta t) \\ -\max(\delta_{min} - \delta_0, -\dot{\delta}_{max}\Delta t) \end{pmatrix}$$



Point	Description	Active constraints
1	Starting point	None
2	Unconstrained optimum	None
3	Best feasible solution in direction of optimum	$\delta_2 \geq 0$
4	Optimum with active set as equality constraints	$\delta_2 \geq 0$
5	Best feasible solution in direction of optimum and final solution	$\delta_1 \geq 0, \delta_2 \geq 0$

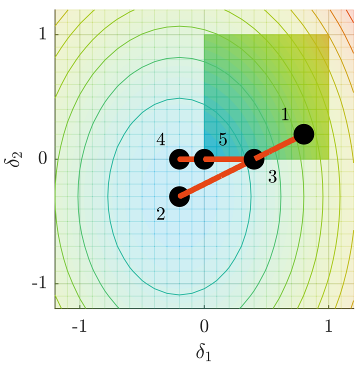


Figure 5: The Active Set Method performed on a hypothetical cost function J with a two-dimensional input space (Constraints: $0 \leq \delta_1 \leq 1$ and $0 \leq \delta_2 \leq 1$, starting point: $(\delta_1, \delta_2) = (0.8, 0.2)$)

When the inequality constraints are treated as equality constraints ($\mathbf{A} = b$ instead of $\mathbf{A} \leq b$), the solution to the optimization problem is given by the following linear system, as long as $\mathbf{Q}$ is a positive definite matrix [29] and $\mathbf{A}$ has full row rank [30]:

$$\begin{bmatrix} \mathbf{Q} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta\delta \\ \lambda \end{bmatrix} = \begin{bmatrix} -c \\ b \end{bmatrix} \tag{4}$$

where λ is known as the vector containing the Lagrange multipliers. Explicit solutions for both the optimal input increment $\Delta\delta$ and Lagrange multipliers λ can be derived algebraically as:

$$\Delta\delta = -\mathbf{Q}^{-1}(\mathbf{A}^T\lambda + c) \tag{5a}$$
$$\text{where } \lambda = -(\mathbf{A}\mathbf{Q}^{-1}\mathbf{A}^T)^{-1}(\mathbf{A}\mathbf{Q}^{-1}c + b) \tag{5b}$$

The values of the Lagrange multipliers are used to determine what constraints to release during the optimization process, and whether or not the solution has already reached its optimum.

Since the calculation of UAV control demands typically needs to be performed several hundred times per second, the optimization used in an INCA controller needs to be as efficient as possible. Based on control allocation research performed by [24] and [27], the optimization method selected for this research is the *Active Set Method*. This method requires similar amounts of computing power as e.g. the Redistributed Pseudo-Inverse method and the Fixed-Point algorithm, yet yields more accurate solutions. It also scales efficiently with larger amounts of actuators, which is validated in Section 8.

A detailed description of the Active Set Method [31] is summarized below and illustrated in Figure 5 for a hypothetical optimization problem with a constrained two-dimensional input space:

Step 1:

Choose a feasible starting point

Step 2:

Determine the *active set* of constraints, i.e. all constraints at which a control command saturates. Redefine the optimization problem using only the active constraints as equality constraints.

$k = 1, 2, \ldots, N$

Step 3:

Calculate the Lagrange multipliers and solution to the redefined problem using Equations 5a and 5b.

Step 4:

If the solution is infeasible:
Correct the solution by taking the maximum relative step to the new solution without losing feasibility and determine the new active set of constraints.

Else if not all $\lambda \geq 0$:
Release the constraint corresponding to the most negative value in λ from the active set of constraints.

Else: The optimal solution has been found.

Step 5:

Repeat from Step 3 with the new active set of constraints while the optimal solution has not been found.

Choosing a suitable starting point for the Active Set Method has a significant effect on the solver's efficiency. In

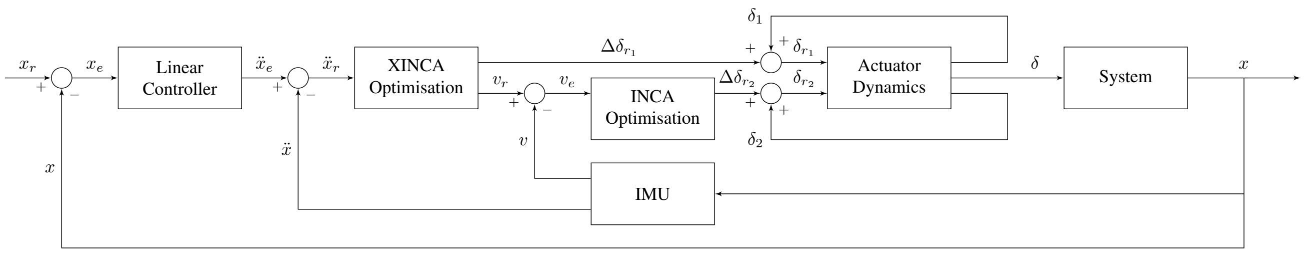


Figure 6: A schematic representation of a XINCA controller (x = state vector, v = virtual input, δ = control input vector)

control allocation, each solution is likely to be in the neighborhood of the solution of the previous time step. The Active Set Method has a relatively low computational cost [31], and since the solution progresses towards the final solution each time step, even when cut off before reaching the optimum to save computational time, the solution will be close to optimal.

5 XINCA

To simplify the outer loop control, hybrid UAVs like quadplanes are often controlled in either a vertical, horizontal or short transitional flight mode. Separating these flight modes however often results in sub-optimal flight control, not always using the most effective or efficient actuators nor making use of redundant actuators in case of actuator saturation. Worse even, actuator sets can even counteract each other. In hover, for instance, a downward pitch to move forward with a non-zero wind will cause a negative lift of the wing counteracting and possibly saturating the hover motors.

To solve this, a new control scheme is proposed, called Extended Incremental Non-linear Control Allocation (XINCA). It is an extension of INCA which takes the specific quadplane outer-loop dynamics into account as additional constraints. As shown in Figure 6, a linear controller on the position errors selects the desired linear reference accelerations. The error between these reference accelerations and measured accelerations then enters the XINCA optimization block. Like the INCA optimization, the XINCA optimization possesses several constrained actuators to achieve this control demand with, albeit these XINCA actuators do not only include physical actuators of the platform, but also some of its attitude angles and its vertical thrust command. In the case of this research, the XINCA output includes the tail pusher rotor command, the vertical thrust command, and the vehicle's pitch and roll commands. The tail rotor command is directly fed to the tail rotor itself. The thrust command and two attitude angle commands serve as input for the inner loop's INCA optimization. The XINCA optimization is also performed with the Active Set Method. Since the effectiveness of the XINCA actuators is also highly dependent on the aircraft's states, it needs to be re-assessed at every iteration. But the resulting controller does not need to differ anymore for any of the flight regimes or flight modes.

6 IMPLEMENTATION

XINCA is implemented in the open-source drone hardware and software platform Paparazzi UAV [32]. The quadplane itself makes use of a *Lisa/MX autopilot* board. Since this board can control a maximum of eight actuators, the two ailerons share one control command, making them respond symmetrically yet in the opposite direction while reducing the computational cost of the INCA optimization.

The INCA module in Paparazzi UAV is based on an INDI module from [25], and used by [13]. It is extended to include seven of the quadplane's eight actuators, and scale the effectiveness of the three actuators that are aerodynamic surfaces, i.e. two separate ruddervators and the combined ailerons. The achieved control is calculated as follows:

$$\begin{bmatrix}\Delta\dot{p} & \Delta\dot{q} & \Delta\dot{r} & \Delta\ddot{z}\end{bmatrix}^T = \mathbf{H}\Delta\delta \tag{6}$$

where $\delta = \begin{bmatrix}\delta_{r_{lf}} & \delta_{r_{rf}} & \delta_{r_{rr}} & \delta_{r_{lr}} & \delta_a & \delta_{r_l} & \delta_{r_r}\end{bmatrix}^T$

The control effectiveness matrix $\mathbf{H}$ is separated into two parts. $\mathbf{H_1}$ accounts for increments in actuator inputs, and $\mathbf{H_2}$ accounts for counter-torque effects during the spin-up of the upwards facing rotors, such that:

$$\mathbf{H} = \mathbf{H_1} + \Delta t\mathbf{H_2} \tag{7}$$

The actuator effectiveness matrix units are either rads^{-2} PPRZ^{-1} or ms^{-2} PPRZ^{-1}, where PPRZ stands for Paparazzi actuator units ranging from -9600 for bi-directional or 0 for mono-directional actuators to 9600. To illustrate INCA's ability to handle inaccurate actuator models because of its incremental nature, only a simple approximation of the actuator effectiveness is used to control the quadplane. This approximation is based on theoretical calculations using estimations of the inertial properties and their actuator positions. The resulting actuator effectiveness matrices are:

$$\mathbf{H_1} = 10^{-3}\cdot \begin{array}{ccccccc c} \delta_{r_{lf}} & \delta_{r_{rf}} & \delta_{r_{rr}} & \delta_{r_{lr}} & \delta_a & \delta_{r_l} & \delta_{r_r} & \\ \left[\begin{matrix} 11 \\ 9 \\ -0.6 \\ -0.8 \end{matrix}\right. & \begin{matrix} -11 \\ 9 \\ 0.6 \\ -0.8 \end{matrix} & \begin{matrix} -11 \\ -9 \\ -0.6 \\ -0.8 \end{matrix} & \begin{matrix} 11 \\ -9 \\ 0.6 \\ -0.8 \end{matrix} & \begin{matrix} 0.15u^2 \\ 0 \\ 0 \\ 0 \end{matrix} & \begin{matrix} 0 \\ 0.11u^2 \\ -0.03u^2 \\ 0 \end{matrix} & \left.\begin{matrix} 0 \\ -0.11u^2 \\ -0.03u^2 \\ 0 \end{matrix}\right] & \begin{matrix} \Delta\dot{p} \\ \Delta\dot{q} \\ \Delta\dot{r} \\ \Delta\ddot{z} \end{matrix} \end{array} \tag{8a}$$

$$\mathbf{H_2} = 10^{-3} \cdot \begin{array}{c} \begin{array}{ccccccc} \delta_{r_{lf}} & \delta_{r_{rf}} & \delta_{r_{rr}} & \delta_{r_{lr}} & \delta_a & \delta_{r_l} & \delta_{r_r} \end{array} \\ \left[\begin{array}{ccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -55 & 55 & -55 & 55 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \end{array} \begin{array}{c} \\ \Delta\dot{p} \\ \Delta\dot{q} \\ \Delta\dot{r} \\ \Delta\ddot{z} \end{array} \tag{8b}$$

where u represents the true airspeed over the aerodynamic control surfaces, which in this work is simplified by the substitution of the forward body velocity since tests are performed in an indoor environment without wind. Negative values are replaced by zero.

Since the actuators do not provide any form of feedback, an estimation of the current actuator deflection is performed each time step. This is done by a first-order approximation with a time constant τ:

$$H_{act} = \frac{K}{\tau s + 1} \tag{9}$$

Each actuator position is estimated as:

$$\delta_{est} = \delta_{prev} + \alpha(\delta - \delta_{prev}) \tag{10}$$
$$\text{where } \alpha = 1 - e^{-\tau \Delta t}$$

The used time constant used for the four upwards facing rotors is 29 s^{-1}. For the control surfaces, an estimation of 100 s^{-1} is used. The optimization parameters in Equations 2a and 2b are chosen as:

$$\mathbf{W}_\tau = diag\begin{bmatrix}100 & 100 & 1 & 1000\end{bmatrix}$$
$$\mathbf{W}_\delta = diag\begin{bmatrix}10 & 10 & 10 & 10 & 1 & 1 & 1\end{bmatrix}$$
$$\gamma = 10000$$
$$\delta_p = \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}^T$$

These values are selected to prioritize pitch and roll and especially thrust over yaw commands while $\mathbf{W}_\delta$ penalizes the use of rotors over aerodynamic surfaces as the latter use less energy. Finally, γ prioritizes achieving the control demand over minimizing control effort. The actuator limits are set to either 0 and 9600 for rotors or -9600 and 9600 for control surfaces, again expressed in PPRZ units.

The XINCA controller works in a similar manner as the INCA controller and is based on an existing outer loop INDI module by [33, 34]. This existing module uses the vertical thrust vector to control the position, by either changing this thrust itself or changing its orientation by pitch or roll increments. It is augmented by including a tail rotor command as its fourth actuator. The control is then calculated as follows:

$$\begin{bmatrix}\Delta\ddot{x} & \Delta\ddot{y} & \Delta\ddot{z}\end{bmatrix}^T = \mathbf{H}\begin{bmatrix}v_r & \delta_{r_t}\end{bmatrix}^T \tag{12}$$
$$\text{where } v_r = \begin{bmatrix}\Delta\theta & \Delta\phi & \Delta T\end{bmatrix}^T$$

The actuator effectiveness highly depends on the current state. At low speeds aerodynamics do not play a great role yet, so it could be calculated as follows:

$$\mathbf{H} = \begin{array}{c} \begin{array}{cccc} \Delta\theta & \Delta\phi & \Delta T & \delta_{r_t} \end{array} \\ \begin{bmatrix} -c\theta c\phi T & -s\theta s\phi T & s\theta c\phi & c\theta \\ 0 & -c\phi T & -s\phi & 0 \\ s\theta c\phi T & -c\theta s\phi T & c\theta c\phi & -s\theta \end{bmatrix} \end{array} \begin{array}{c} \\ \Delta\ddot{x} \\ \Delta\ddot{y} \\ \Delta\ddot{z} \end{array} \tag{13}$$

where s and c represent the sine and cosine functions respectively, and T represents the vertical specific-force vector, which is estimated by taking the vertical body acceleration and subtracting the gravitational acceleration:

$$T = \ddot{z} - g \tag{14}$$

When flying at higher velocities, however, the quadplane will start to behave more like a fixed-wing aircraft. The controller should start using the wings to generate lift instead of the hover motors, and as a positive angle of pitch leads to a positive angle of attack on the main wing, one term is added to the actuator effectiveness matrix as follows:

$$\mathbf{H} = \begin{array}{c} \begin{array}{cccc} \Delta\theta & \Delta\phi & \Delta T & \delta_{r_t} \end{array} \\ \begin{bmatrix} -c\theta c\phi T & -s\theta s\phi T & s\theta c\phi & c\theta \\ 0 & -c\phi T & -s\phi & 0 \\ c\phi\left(s\theta T - \dfrac{C_{L_\alpha}\rho u^2 S}{2m}\right) & -c\theta s\phi T & c\theta c\phi & -s\theta \end{bmatrix} \end{array} \begin{array}{c} \\ \Delta\ddot{x} \\ \Delta\ddot{y} \\ \Delta\ddot{z} \end{array} \tag{15}$$

where C_{L_α} is the change in lift per change in angle of attack, ρ is the air density, u is the true airspeed, S is the wing surface area, and m is the platform's mass. In Equations 2a and 2b the XINCA optimization parameters are chosen as:

$$\mathbf{W}_\tau = diag\begin{bmatrix}10 & 10 & 1\end{bmatrix}$$
$$\mathbf{W}_\delta = diag\begin{bmatrix}10 & 10 & 100 & 1\end{bmatrix}$$
$$\gamma = 10000$$
$$\delta_p = \begin{bmatrix}0 & 0 & 0 & 0\end{bmatrix}^T$$

$\mathbf{W}_\tau$ prioritizes pitch and roll over thrust demands since an unstable attitude can be more dangerous than a controlled descent. Moreover, the pitch is used in lift generation. $\mathbf{W}_\delta$ penalizes the use of pitch and roll and especially thrust commands compared to using the tail rotor, and γ prioritizes achieving the control demand over minimizing the control effort. The maximum pitch and roll angles are set to 10°, the vertical thrust limits to -9.0 and 9.0 ms^{-2}, and the tail rotor's limits to 0 and 9600 PPRZ units. To prevent the tail rotor from hitting the ground, it is completely shut off for altitudes below 0.5 m by setting its effectiveness to zero.

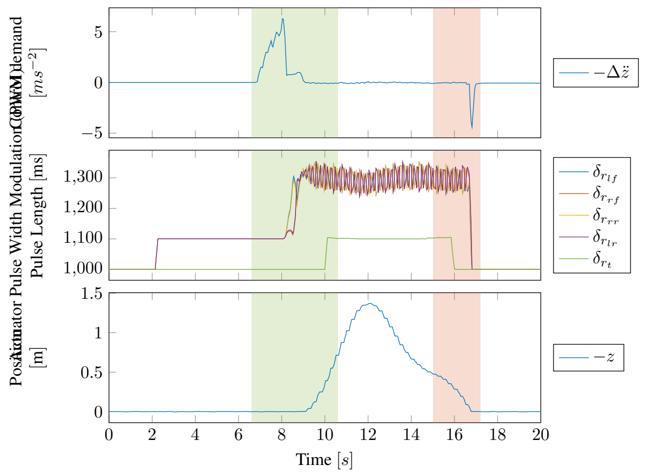


Figure 7: Simulation of vertical quadplane takeoff and landing using XINCA and INCA with five actuators
🟩 = takeoff, 🟥 = landing

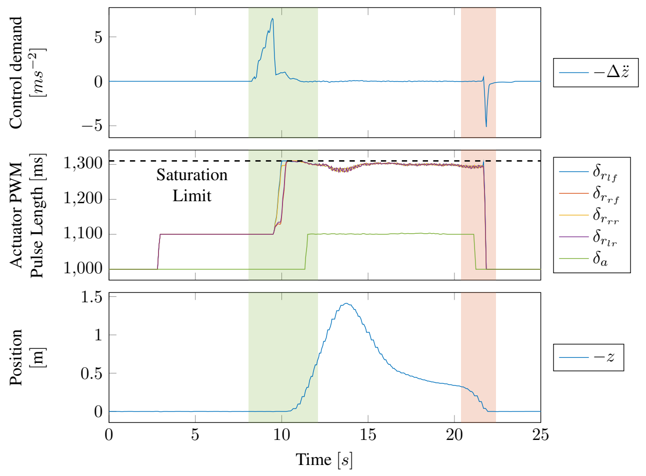


Figure 8: Simulation of a vertical quadplane takeoff and landing with actuator saturation occurring at an actuator PWM pulse length of 1310 ms using XINCA and INCA with five actuators
🟩 = takeoff, 🟥 = landing

7 Flight Simulations

To prove XINCA's performance, several simulations are performed. These are executed within the Paparazzi UAV software to fully assess the performance of the actual code that will also fly onboard the quadplane.

Figure 7 shows a simple simulation of a quadplane taking off (green) and landing (red). The top plot shows the control demand the INCA controller aims to achieve. The middle plot shows the resulting actuator commands expressed in PWM pulse length. The bottom plot shows the height profile of the flight.

To assess how well the INCA controller handles actuator

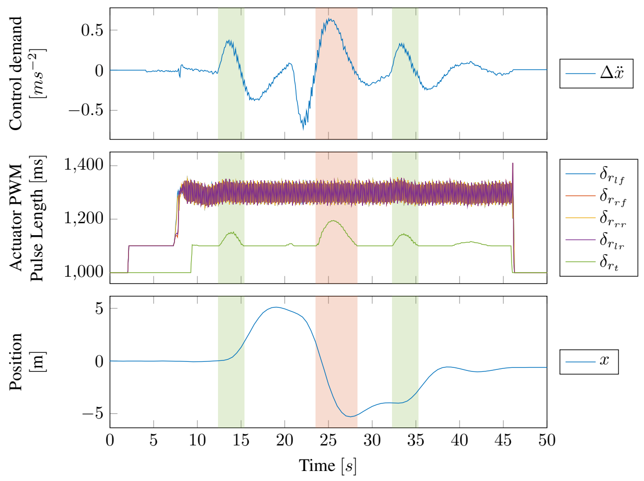


Figure 9: Simulation of forwards and backward flight using XINCA with both pitch increments and tail rotor inputs and INCA with five actuators
🟩 = forward acceleration by tail rotor, 🟥 = tail rotor braking

saturation, a second simulation is performed with an artificial upper actuator limit slightly higher than the nominal throttle level needed for hovering. The result can be seen in Figure 8, which shows that saturation occurs during takeoff. The INCA controller achieves stable flight since its pitch and roll commands are prioritized above its thrust and especially yaw commands.

In the third simulation, the UAV moves forward and backward. Figure 9 shows the control demand and position in the x-direction. The green and red areas show where the tail rotor is being activated by the XINCA controller for acceleration and reducing backward speed respectively. The tail rotor is first activated to accelerate forward. The UAV then uses pitch increments to brake and accelerate backward, after which it activates the tail rotor again twice to brake and move forward again. Finally, it slows down using pitch increments and lands.

Two last simulations are performed to illustrate the benefits of using XINCA over conventional outer loop control methods. Both simulate forward flight of the quadplane and compare a traditional INDI outer-loop controller [33, 34] with the novel XINCA controller. Using the main wing's aerodynamic properties in combination with the quadplane's pitch angle and forward velocity, an estimation is made of the wing-induced lift force. Figure 10 shows the actuator commands, pitch angle, and lift force for both simulations. The most evident difference can be seen in the pitch angles. Where the INDI controller aggressively pitches forward to achieve forward acceleration, the XINCA controller proves to be able to minimize this negative pitch by using its tail rotor. This difference is reflected in the lift force estimations

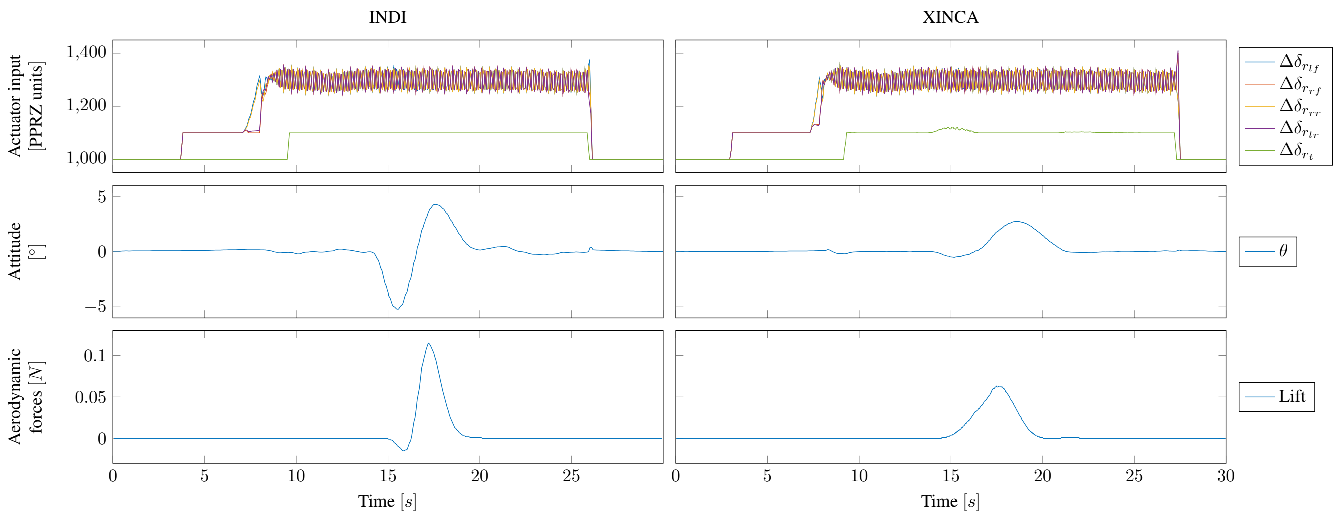


Figure 10: Flight data comparison of forward flight simulation with INDI and XINCA

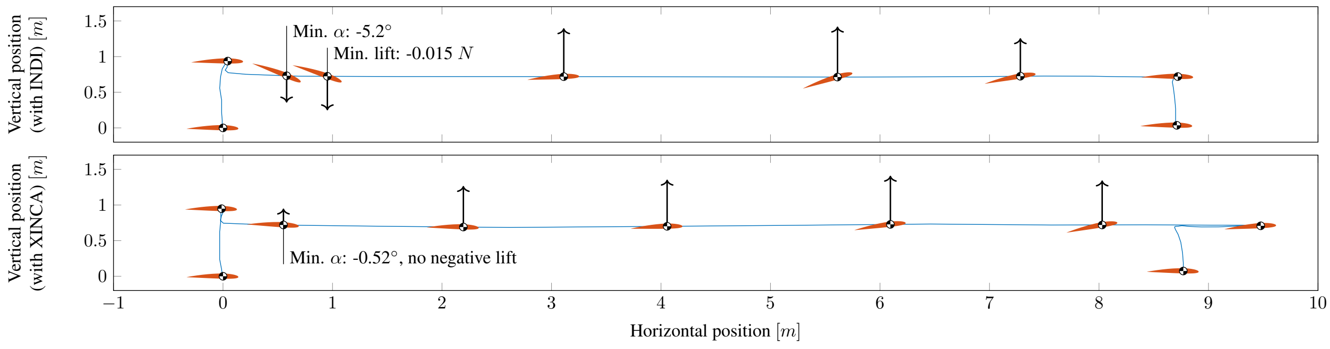


Figure 11: Flight profile comparison of forward flight simulation with INDI and XINCA showing wing-induced lift estimations. Note that illustrated angles of attack are magnified and force vectors are scaled for readability

shown in Figure 11, where the XINCA controller manages to completely avoid the negative lift caused by pitching forward. The INDI controller does inflict some negative lift. At higher wind speeds this negative lift can become very significant and result in an important loss of altitude.

8 Flight Experiments

The XINCA controller was tested in real flight tests of the TU Delft quadplane shown in Figure 1. The flight tests are performed in the *Cyberzoo*, which is equipped with an optical position tracking system for precise vehicle positioning.

During initial attempts to fly the Quadplane with both the INCA and XINCA optimizations, the 32-bit STM32-F4 Central Processing Unit (CPU) processor running at 266 MHz could get overloaded. The first measure to reduce the computational cost of the controllers is to run the optimizations of both the inner and outer loops only once every second iteration of the autopilot, which runs at a cycle frequency of 512 Hz. A system monitoring module in Paparazzi has been used to estimate the autopilot's CPU loads with different configurations using this reduced optimization frequency. These configurations include a combination of the INCA controller with a lower cost outer loop controller, a combination of a lower cost inner loop quadcopter controller with the XINCA controller, and a combination of both the INCA and XINCA controllers. For configurations using the INCA controller, the amount of INCA actuators is varied to determine its effect on computational cost. The results of these measurements can be seen in Table 1. These measurements are obtained on the quadplane itself, yet without flying.

Because of the Active Set Method, the numbers clearly show a quasi-linear correlation between the number of actuators and the CPU load, and that the configuration with both INCA and XINCA does indeed demand a lot of the autopilot's computing power. The fact that the maximum recorded CPU load is still well below 100% can be explained by the fact that the optimization schemes only run once every two cycles, resulting in an average load under 100%. The actual load during one optimization cycle might however re-

	Inner loop:	INCA	Other	INCA
	Outer loop:	Other	XINCA	XINCA
INCA Actuators	4	38%	32%	48%
	5	46%		54%
	6	54%		62%
	7	62%		71%
	8	74%		83%

Table 1: CPU load estimations for different inner and outer loop controllers with different numbers of INCA actuators

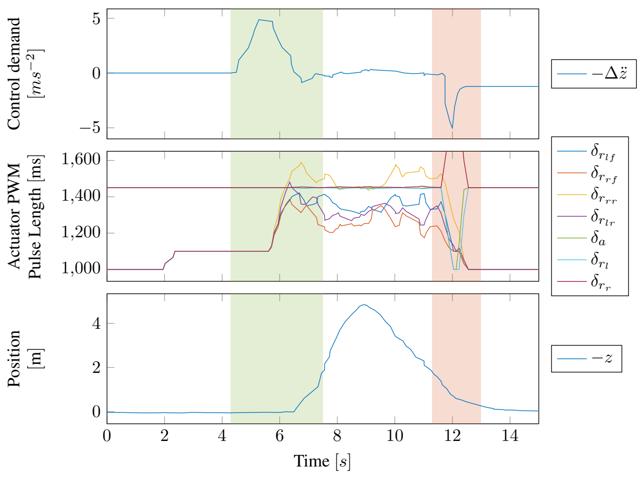


Figure 12: Stable quadplane takeoff and landing using INCA with seven actuators
🟩 = takeoff, 🟥 = landing

quire significantly more computing power, resulting in unpredictable behavior of the quadplane. Especially some time-critical processes need to be re-evaluated to perform well under high CPU load. Ideally, the quadplane's autopilot board is to be replaced by one with sufficient computing power. For this research, however, flight tests will be performed with either both INCA and XINCA without any control surfaces, or INCA with all inner loop actuators and a low-cost outer loop controller.

The first test flight aims to confirm that the INCA controller chooses suitable actuators during flight. All inner loop actuators are included in the optimization, so a low-cost outer loop controller is used for this test. Since the Cyberzoo's confined space only allows for low-velocity testing, the controller is not expected to allocate a significant amount of control to the control surfaces. The results in Figure 12 confirm this. They show varying inputs for the quadplane's upwards facing rotors, due to a slight asymmetrical configuration, but successfully ensure a stable takeoff and landing. As soon as the Quadplane touches down, the ground forces result in unreachable control demands. This causes the rotors to saturate at their minimum values, after which the control surfaces are saturated as well in a maximum effort to reach the setpoint.

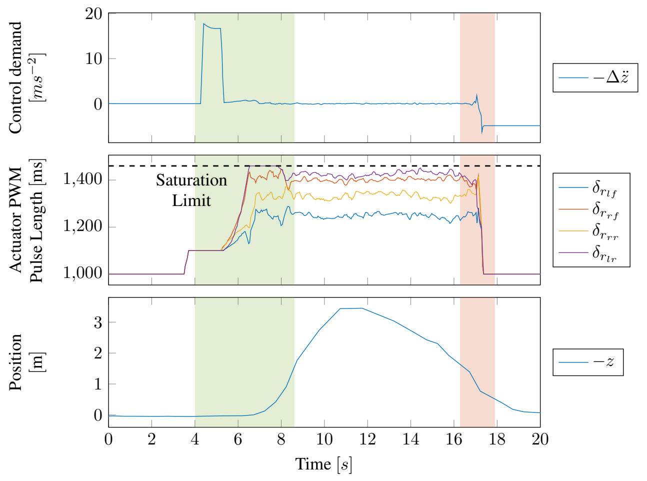


Figure 13: Stable quadplane takeoff and landing with actuator saturation occurring at an actuator PWM pulse length of 1460 ms using INCA with five actuators
🟩 = takeoff, 🟥 = landing

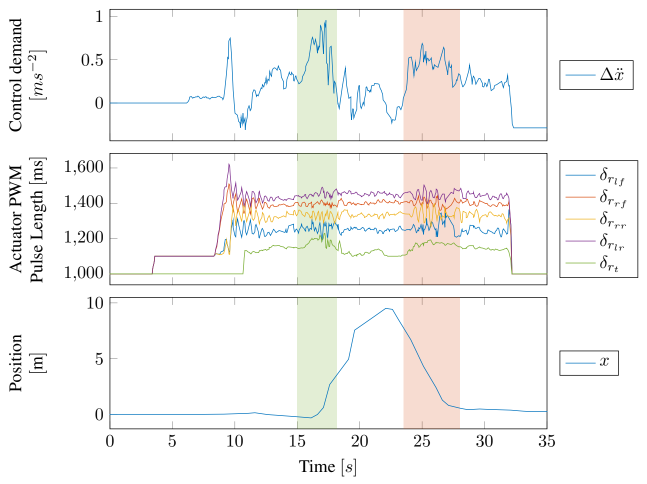


Figure 14: Forwards and backwards quadplane flight using XINCA with both pitch increments and tail rotor inputs and INCA with five actuators
🟩 = forward acceleration by tail rotor, 🟥 = tail rotor braking

The difference in actuator inputs between different rotors seen in the first flight can be exploited in the second, where INCA's resilience against actuator saturation is being put to the test. The saturation level is chosen in such a way that one actuator especially saturates, in this case, $\delta_{r_{lr}}$. Like with its corresponding simulation, Figure 13 shows that INCA prioritizes its pitch and roll commands above its thrust and especially yaw commands, resulting in slower but stable takeoff. Saturating actuators does result in the INCA optimization having to perform more iterations before it reaches its optimum since the Active Set Method has to explore the edges of the actuator input space in multiple steps. This eventually results in a higher computational load. This test is therefore performed with the INCA controller using only four actuators and a low-cost outer loop controller.

The final flight is the one where the novel XINCA module is being tested. For this flight, the quadplane is controlled by both the INCA and XINCA controllers that together allocate control to a total of five rotors. The flight consists of a takeoff, forward flight, backward flight, and landing. Figure 14 shows that the quadplane effortlessly manages to perform this longitudinal maneuver. Peaks in the tail rotor command show that this actuator is indeed used for both forward acceleration and backward braking as expected.

9 Conclusions and Recommendations

During both the simulations and the actual test flights, it was confirmed that the INCA controller chooses suitable actuators and achieves stable flight even in the case of actuator saturation. Prioritizing certain control demands over others successfully ensures stable flight when saturation occurs. Furthermore, the XINCA controller seamlessly takes the fixed-wing constraints into account in all flight phases without needing to switch modes. Furthermore, it proves to not require very detailed models of its controlled vehicle, and the Active Set Method makes it suitable for real-time optimization at high frequencies. Recalculation of the actuator's effectiveness at every time step results in high automated adaptability to changing states and conditions to ensure efficient flight control, using the most suitable and efficient actuators available. When optimizing commands for too many actuators, however, this INCA controller is not efficient enough to be used on the TU Delft Quadplane in its current hardware configuration. Allocating control to seven actuators while using a low-cost outer loop controller is at the edge of its computational capacity. Future research on this specific platform, therefore, requires hardware upgrades to achieve more computing power.

Finally, the novel XINCA controller is capable to perform an optimization in the outer control loop by combining attitude angle commands as well as direct actuator commands. This method eliminates the inefficient use of separated flight modes while avoiding pitfalls like negative main wing lift in hover. This can contribute to a safer, more efficient, and therefore greener future of human aerial transportation.

Future research on the application of INCA on hybrid vehicles like the quadplane and the application of XINCA in general should focus on their performance during level flight, as this has not been sufficiently addressed during this work. Outdoor flights should serve two main research objectives. One objective would be to assess how the quadplane allocates more control to its aerodynamic control surfaces as soon as it has an amount of forward airspeed making them more effective. The other objective focuses on XINCA, assessing its capabilities to adapt to the different dynamics of a hovering quadplane and one in forward flight.

References

[1] Elisabeth Bumiller and Thom Shanker. War evolves with drones, some tiny as bugs. *The New York Times*, Jun 2019.

[2] Amarjot Singh, Devendra Patil, and SN Omkar. Eye in the sky: Real-time drone surveillance system (dss) for violent individuals identification using scatternet hybrid deep learning network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[3] A. Momont. Drones for good. Master's thesis, Delft University of Technology, 2014.

[4] Lilium, 2020.

[5] U.M. Rao Mogili and B.B.V.L. Deepak. Review on application of drone systems in precision agriculture. *Procedia Computer Science*, 133:502–509, 2018.

[6] Elisabeth Bumiller and Thom Shanker. War evolves with drones, some tiny as bugs. *The New York Times*, Jun 2019.

[7] Sasanka Madawalagama, Niluka Munasinghe, S Dampegama, and L Samarakoon. Low cost aerial mapping with consumer-grade drones. In *37th Asian Conference on Remote Sensing*, pages 1–8, 10 2016.

[8] Zhang Daibing, Wang Xun, and Kong Weiwei. Autonomous control of running takeoff and landing for a fixed-wing unmanned aerial vehicle. *2012 12th International Conference on Control Automation Robotics & Vision (ICARCV)*, 2012.

[9] Marco Palermo and Roelof Vos. Experimental aerodynamic analysis of a 4.6%-scale flying-v subsonic transport. *AIAA Scitech 2020 Forum*, May 2020.

[10] D. Jin Lee, Byoung-Mun Min, Min-Jea Tahk, Hyochoong Bang, and D.h Shim. Autonomous flight control system design for a blended wing body. *2008 International Conference on Control, Automation and Systems*, 2008.

[11] Bai Zhiqiang, Liu Peizhi, Wang Jinhua, and Hu Xiongwen. Simulation system design of a uav helicopter. *2011 International Conference on Electric Information and Control Engineering*, 2011.

[12] Teppo Luukkonen. Modelling and control of quadcopter, Aug 2011.

[13] E.J.J. Smeur, D.C. Höppener, and C. De Wagter. Prioritized control allocation for quadrotors subject to saturation. *International Micro Air Vehicle Conference and Flight Competition (IMAV)*, 2017.

[14] Jacob Apkarian. Attitude control of pitch-decoupled vtol fixed wing tiltrotor. *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2018.

[15] Ryuta Takeuchi, Keigo Watanabe, and Isaku Nagai. Development and control of tilt-wings for a tilt-type quadrotor. *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, 2017.

[16] Christophe De Wagter and Ewoud J.J Smeur. Control of a hybrid helicopter with wings. *International Journal of Micro Air Vehicles*, 9(3):209–217, Nov 2017.

[17] Matthew E. Argyle, Jason M. Beach, Randal W. Beard, Timothy W. Mclain, and Stephen Morris. Quaternion based attitude error for a tailsitter in hover flight. *2014 American Control Conference*, 2014.

[18] Danial Sufiyan Bin Shaiful, Luke Thura Soe Win, Jun En Low, Shane Kyi Hla Win, Gim Song Soh, and Shaohui Foong. Optimized transition path of a transformable hovering rotorcraft (thor). *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2018.

[19] Janith Kalpa Gunarathna and Rohan Munasinghe. Development of a quad-rotor fixed-wing hybrid unmanned aerial vehicle. *2018 Moratuwa Engineering Research Conference (MERCon)*, 2018.

[20] Jian Zhang, Zhiming Guo, and Liaoni Wu. Research on control scheme of vertical take-off and landing fixed-wing uav. *2017 2nd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, 2017.

[21] David Orbea, Jessica Moposita, Wilbert G. Aguilar, Manolo Paredes, Rolando P. Reyes, and Luis Montoya. Vertical take off and landing with fixed rotor. *2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, 2017.

[22] Ma Tielin, Yang Chuanguang, Gan Wenbiao, Xue Zihan, Zhang Qinling, and Zhang Xiaoou. Analysis of technical characteristics of fixed-wing vtol uav. *2017 IEEE International Conference on Unmanned Systems (ICUS)*, 2017.

[23] Gerardo Flores and R. Lozano. Lyapunov-based controller using singular perturbation theory: An application on a mini-uav. *2013 American Control Conference*, 2013.

[24] A.R.J. Stolk. Minimum drag control allocation for the innovative control effector aircraft. Master's thesis, Delft University of Technology, 2017.

[25] Ewoud J. J. Smeur, Qiping P. Chu, and Guido C. H. E. de Croon. Adaptive Incremental Nonlinear Dynamic Inversion for Attitude Control of Micro Aerial Vehicles. *Journal of Guidance, Control, and Dynamics*, 39(3):450–461, March 2016.

[26] Joseph Horn. Non-linear dynamic inversion control design for rotorcraft. *Aerospace*, 6(3):38, 2019.

[27] D.C. Höppener. Actuator saturation handling using weighted optimal control allocation applied to an indi controlled quadcopter. Master's thesis, Delft University of Technology, 2017.

[28] Ola Härkegård. Dynamic control allocation using constrained quadratic programming. *AIAA Guidance, Navigation, and Control Conference and Exhibit*, May 2002.

[29] Henri P. Gavin and Jeffrey T. Scruggs. Constrained optimization using lagrange multipliers. *CEE 201L. Uncertainty, Design, and Optimization*, 2020.

[30] Tor A. Johansen and Thor I. Fossen. Control allocation—a survey. *Automatica*, 49(5):1087–1103, 2013.

[31] O. Harkegard. Efficient active set algorithms for solving constrained least squares problems in aircraft control allocation. *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, 2002.

[32] Pascal Brisset, Antoine Drouin, Michel Gorraz, Pierre-Selim Huard, and Jeremy Tyler. The paparazzi solution. In *MAV 2006, 2nd US-European Competition and Workshop on Micro Air Vehicles*, pages 1–15, Sandestin, United States, October 2006.

[33] E.J.J. Smeur, G.C.H.E. de Croon, and Q. Chu. Cascaded incremental nonlinear dynamic inversion for mav disturbance rejection. *Control Engineering Practice*, 73:79–90, 2018.

[34] Ewoud J. J. Smeur, de G.C.H.E de Croon, and Qiping Chu. Gust disturbance alleviation with incremental nonlinear dynamic inversion. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 5626–5631, Daejeon, South Korea, December 2016. IEEE/RSJ, IEEE.

Position controller for a flapping-wing drone using UWB

Guillermo González*, Guido C.H.E de Croon, Diana Olejnik and Matěj Karásek
Delft University of Technology, Mekelweg 5, Delft

ABSTRACT

This paper proposes an integral approach for accurate ultra wide band indoor position control of flapping wing micro air vehicles. Three aspects are considered to reach a reliable and accurate position controller. The first aspect is a velocity/attitude flapping-wing model for drag compensation. The model is compared with real flight data and shown to be applicable for more than one type of flapping wing drone. The second improvement regards a battery-level dependent thrust control. Lastly a characterisation of ground effects in flapping-wing flight is obtained from hovering experiments. The proposed controller improves position control by a factor $\sim$ 1.5, reaching a mean absolute error of 10cm for position in x and y, and 4.9cm for position in z.

1 INTRODUCTION

The fact that drones are becoming increasingly popular is intimately related with the development of more sophisticated automation resources. In the case of drones, weight, processing speed, and energy consumption are critical aspects to accomplish fully autonomous flights. Therefore, elements such as motors, micro-processors, memory units and batteries need to be continuously improved to meet these requirements. These conditions have fostered the development of flapping wing micro-air vehicles (FWMAV). This type of UAV is inspired on the flight of birds and insects, and has become attractive in the field of small-scale micro-air vehicles, since it provides both the ability of hovering and flying into any direction. After the foundational work from Ellington [1] and Dickinson [2], many experiments have been carried out in order to get an optimal physical design of FWMAVs, as well as velocity and position controllers to keep a stable flight.

Regarding position feedback, the most common option is an indoor positioning system (IPS). Some of the technologies used for for IPS are Wi-fi, Radio Frequency Identification (RFID), and infrared (IR) motion tracking systems. Wi-fi and RFID can be found for tracking mobile devices, but not specifically for drones [3]. IR is used for indoor tracking of drones (e.g. VICON/OptiTrack), but it is an expensive option and its accuracy can be affected by lighting conditions [4].

Another option for IPS is ultra wide band (UWB) which was already defined in 2006, but just recently started to gain popularity. UWB is able to transmit in nano-second scale periods. Thus it allows excellent timing for signal arriving, which translates to centimeter level accuracy. Its low spectral density reduces the interference with other RF devices. However, it also has some drawbacks like high computational and memory capacity of the controller and its vulnerability to multi-path effect when signals bounce with the physical boundaries of the environment [5].

Inertial navigation systems (INS) are usually added in indoor environments to complement IPS. Usually INS systems employ micro-electro mechanical systems (MEMS) or inertial measurement units (IMU). Furthermore, data fusion is also applied for better accuracy, by means of a Kalman filter. Hence many combinations of integrated systems have been proposed. For example a GPS/UWB/MEMS navigation system with Kalman filter [6] and INS/UWB system based on a fuzzy adaptive Kalman filter [7]. Although most of these implementations provide a basis for designing an autonomous position controller, it is important to keep in mind they are specifically designed either for fixed-wing or quadrotor UAVs. In the case of flapping wing drones some extra constraints shall be considered like the physical limitations in terms of payload and energy consumption, or the noise influence in IMU measurements due to high frequency mechanical vibration [8]. To overcome these circumstances, the existing solutions must be adapted, leaving a potential research development for position controllers of FWMAVs.

One of the challenges of working with FWMAVs comes from the aerodynamics of the system. A reliable model of the dynamics can significantly increase the performance and accuracy. Nevertheless, most aerodynamic models for flapping wings require extensive system identification techniques for numerous parameters, whose values only remain valid for a specific drone. A widespread alternative is the use of quasi-steady models where force coefficients are obtained either from experimental data or from theoretical principles [9]. The control strategy proposed in this paper follows a simpler solution where the model is obtained by directly averaging aerodynamic parameters as functions in terms of the body velocities. Albeit the method may be considered just a rough approximation compared to quasi-steady models, it tends to be more practical since just a reduced amount of parameters is required. When validating averaging parameters with real flight data, several authors address the issues of un-

*Email address(es): g.gonzalezarchundia@student.tudelft.nl, g.c.h.e.decroon@tudelft.nl, d.a.olejnik@tudelft.nl, matej@flapper-drones.com steady flight due to mechanical vibrations [10, 11]. However just some models follow an approach based on drag compensation. Within the few cases where drag compensation models are implemented, most of them are linear models, which are only valid for a limited range of velocities [8, 12].

Similar to most MAVs, FWMAVs are also prone to short flights due to the limited size of the batteries they are able to carry. Thus it is common that voltage will significantly change during flight, affecting as well the required throttle level for hovering [13]. This condition poses a challenge in height control, for both reaching and keeping a specific position along the z axis.

Another issue affecting height control is the ground effect. This phenomenon particularly occurs when the drone is flying close to the ground [14]. An extra thrust is generated because the wind currents underneath the drone bounce against the ground, causing the drone to behave like sitting on a cushion of air [15]. Thus, an extra lift is generated, causing the output thrust to be higher than the input thrust.

This paper addresses the three aforementioned challenges of FWMAVs. Sections 2, 3 and 4 provide the background for the implementation. Section 5 describes how averaging parameters are applied for drag compensation. Section 6 discusses the issues of changing voltage and ground effects. Lastly, Section 7 shows the results on how the transient response is improved in any of the three axes: forward (x-axis), sideways (y-axis) and vertical (z-axis).

2 Experimental setup

Since the work on this project relies mainly on feedback through UWB, the experimental setting should be a grid where UWB receivers are strategically located in the vertices and the transceiver is mounted on the FWMAV. This way, the drone flies inside the volume of a cube bounded by the position of the anchors. A similar set-up is used in [16], where the eight anchors on the corners optimise the possibility for the drone to have line-of-sight at least with one anchor. Such a grid was set-up at the Cyberzoo (shown in Figure 1), the flight arena of the TU Delft Faculty of Aerospace, which is known as the main facility of the university for performing tests on drones, and has a size of $10m \times 10m \times 7m$. A reliable choice for UWB sensor is the Decawave DWM1000, as it has proven to give successful results for UAV tracking [17]. The UWB sensors are used as the anchors of the IPS and are set to work using a time difference of arrival (TDOA) algorithm.

In order to have a reference for the UWB measurements, a motion tracking system is used for measurements of position and rotations. It consists of 12 OptiTrack Prime 17W motion tracking cameras (set to resolution 1664 px $\times$ 1088 px, 50 fps) and has proven to deliver accurate results for MAV test flights [18]. When working with the motion tracking system, the drone was equipped with four retro-reflective markers placed on the landing gear of the drone, and the UWB sensor was placed at the top to optimise direct line of sight

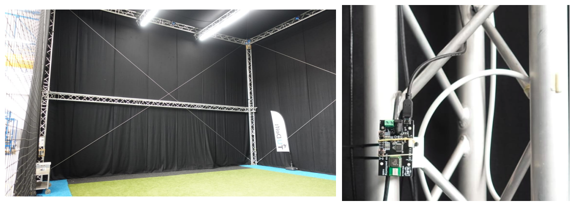


Figure 1: Cyberzoo structure where the UWB anchors are placed (left) and detail of UWB anchor mounted on the structure (right)

(Figure 2).

About the flapping-wing drone, referred in this paper as Flapper, is a design from the company Flapper-drones [19]. It is a 102g tailless FWMAV with a wingspan of 49cm, able to keep flapping frequencies up to 12 Hz when hovering. The on-board processing hardware consists of a Crazyflie Bolt autopilot board, including an IMU with 3-axis accelerometer/gyroscope (BMI088). The data link between the autopilot and the ground station is done with a Crazyradio PA (also from Bitcraze), which is a USB radio dongle based on nRF24LU1+ from Nordic Semiconductor. The system can be powered with a 300-mAh two-cell 7.4V LiPo battery, reaching a flight time between 4-6 minutes, depending on the internal resistance of the battery.

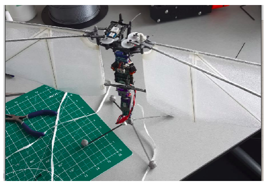


Figure 2: Flapper drone equipped with IR markers and UWB sensor

3 Control loop

For autonomous flight, the flapping-wing drone uses a cascaded PID-controller, based on the structure presented by [20]. The controller consists of three loops. The output provides the control signals for the motors involved in the flight dynamics of the drone: two brush-less motors in charge of the flapping frequency for thrust and roll, one servo-motor that modifies the angle of the dihedral for pitch, and another servo-motor that changes the deflection of the wings for yaw.

Regarding the structure of the cascaded loops, the innermost loop is in charge of attitude rate and runs at 500Hz. The intermediate loop also runs at 500Hz and is in charge of attitude control. The outer loop is the position/velocity control, which runs at 100Hz and can take either position or velocity commands.

Another remark is the trimming values for servos in charge of pitch, roll and yaw. This procedure must be done in order to get the proper behaviour from the controller. For trimming, one should fly the drone manually until it hovers at a steady position. The trim values are the pitch, roll and yaw commands given to keep the drone hovering. The values are specific for each flapping-wing drone, as they change depending on manufacturing factors. Thus it is important to properly set the appropriate trimming values before going further with any flight test.

4 State estimation

The state estimation is done by means of an Extended Kalman Filter (EKF) which fuses the measurements of the IMU and the UWB positioning system. The model of the EKF is the same as the one proposed by [21], which considers a nine-dimensional state vector defined as:

$$S = \begin{bmatrix} x_E & v_B & d \end{bmatrix}^T \tag{1}$$

Where x_E is the position vector in global frame, v_B is the velocity vector in body frame and d is an attitude error vector, where the error is defined as the difference between the last measured attitude and the current attitude. The purpose for using attitude errors instead of the conventional Euler angles is to simplify the state prediction equations because it only considers the increments in roll, pitch and yaw. A more detailed explanation on the implementation of the filter is given in [21].

The main advantage of fusing UWB with IMU through the Kalman filter is the attitude correction to account for the drift in pitch and roll caused by sensor noise and bias. For instance, when the Flapper is left standing up, the Kalman filter resets correctly the roll and pitch to zero when UWB measurement are coming in. On the other side, a complementary filter, which relies only on IMU data, will converge to a certain drift and thus propagates it through time.

One last adjustment done to the EKF aims to diminish the detrimental effects on position estimation caused by multipath UWB signals. Specially when going close to the ground, the position estimates tend to drift for more than 20 cm. A way to account for the wrong measurements when flying close to the ground is to use a variable sensor noise value R_{uwb} for measurements from UWB, rather than a constant value. Consequently, the sensor noise is defined as:

$$R_{uwb} = \lambda(\hat{z})\sigma_{uwb}^2 \tag{2}$$

Where $\lambda(\hat{z})$ is a factor dependent on the estimated height $\hat{z}$, in meters. For the implementation, $\lambda = 0.5$, for $z \geq 1$; $\lambda = 1.5 - \hat{z}$, for $0.5 < \hat{z} < 1$; and $\lambda = 1$, for $z \leq 0.5$. The effect of variable sensor noise in position estimation was tested by placing the drone static on the ground, at the origin (0,0,0) of the UWB IPS. The plots from Figure 3 show the estimated position (est) in contrast to the real position (cmd), Both values were sampled at 50Hz, during 8 seconds. Using the variable sensor noise R_{uwb} significantly increases the accuracy in estimation approximately by 0.2m in the three axes.

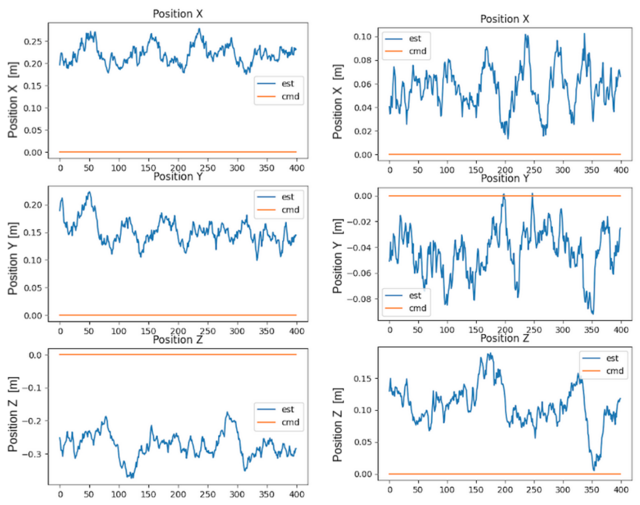


Figure 3: Effects in position estimation before using variable sensor noise (left) and after implementing it (right).

Once the variable sensor noise term was implemented into the EKF, several flight paths were tested (linear, square, rhombus and hexagonal paths). The measurements from UWB using a TDOA algorithm proved to be reliable enough for position estimation. The mean absolute error obtained in all cases remained between 8-10 cm, when compared to the measurements from the Optitrack system.

5 Drag compensation

The first technique for yielding a better control strategy for FWMAVs was to obtain a model for drag compensation. Since the drag is typically neglected in aerodynamic models, there tends to be an offset between the commanded velocity and the velocity output by the controller. Usually a feed forward term is used to compensate for these drag effects. Following the structure of the controller in Section 3, the velocity loops provides the input for the attitude loop. Thus, feed forward is modelled as a function of velocity:

$$\theta_{FF} = f(v_{xE}) \tag{3}$$

$$\varphi_{FF} = f(v_{yE}) \tag{4}$$

Where Eq. (3) is the feed forward term for pitch and Eq. (4) is for roll. In order to derive such a model, a system identification experiment is proposed based on the supplementary materials from [22], where step inputs in roll and pitch are given to the flapping-wing drone and then the transient response is recorded. The experiments are done using manual flight via a Frsky RC-controller where the pitch and roll step inputs are pre-programmed. The tracking data is recorded with the motion tracking system mentioned in Section 2.

Figure 4 is given as an example of the transient response in velocity obtained for different step-inputs in pitch (a similar response is obtained for roll angles). Due to limitations

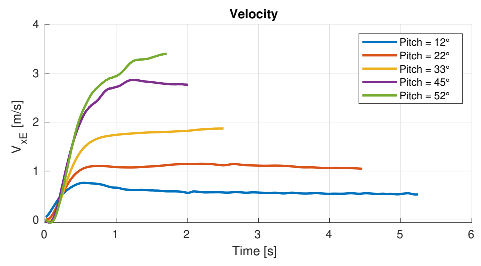


Figure 4: Velocities attained at different pitch step inputs

on the size of the flight arena, only for the small angles (below 45°) it is possible to reach the steady-state velocities. For higher angles, a common technique to reach the steady-state velocities is to perform wind-tunnel experiments [23]. In this case, an alternative method is used where a non-linear regression model is obtained for the smaller angles and then it is used to extrapolate the data at higher angles. From figure 4, it can be appreciated that the transient response approximates to the behaviour of a first-order system. Hence, the equation for the transient response is modelled as an exponential function of the form:

$$v(t) = a - be^{-ct+d} \tag{5}$$

The parameters a, b, c and d from Eq. (5) have to be initialised with certain values, depending on the shape of the curve obtained from the measured data points. The final steady-state values obtained from the extrapolation model strongly depend on how many data points are considered for the regression model. Hence, Table 1 shows the absolute error between the last measured velocity for each angle, and the respective value calculated from the extrapolation model. For the angles of 30° or less, this velocity approximates to the steady-state velocity. For angles of 45° or above, the last measured velocity is used as ground truth equilibrium velocity, although it is still part of the transient response.

Final velocity error at different pitch angles [m/s]					
Data taken for regression	**12°**	**22°**	**33°**	**45°***	**52°***
10%	0.124	0.0752	0.1842	0.2641	0.276
30%	0.0318	0.0459	0.0347	0.1899	0.1259
50%	0.0166	0.0636	0.015	0.2048	0.0038
70%	0.0183	0.0632	0.0282	0.1379	0.0402
90%	0.0142	0.0561	0.0289	0.0857	0.0185

Final velocity error at different roll angles [m/s]					
Data taken for regression	**12°**	**22°**	**33°**	**47°***	**52°***
10%	0.1007	0.1479	0.0391	0.2972	0.3028
30%	0.10	0.1216	0.0452	0.2303	0.2366
50%	0.0262	0.1427	0.0325	0.0499	0.0849
70%	0.0234	0.0802	0.0378	0.0486	0.095
90%	0.0268	0.0247	0.0261	0.0266	0.0757

Table 1: Extrapolation errors taking different amounts of data for regression model. *Indicates that for those angles the final velocity is not considered as ground truth. Hence these errors are overestimated.

As expected, results from Table 1 show on the angles of accurate ground truth (12°,22° and 33°) that the overall accuracy of the extrapolation model, for both pitch and roll, increases as more data points are taken for the regression. Individually, for each angle, accuracy remains approximately the same with 50% of the data points or more. For illustration purposes, Figure 5 shows the predicted steady-state velocity for each angle when taking 70% of the data points for the regression model. Nevertheless, the real steady-state values used were the ones obtained taking all of the data points since those ones result in the lowest error.

Once the steady-state velocity for each angle is available, the pitch-velocity and roll-velocity models can be obtained. Given the mapping of the steady-state velocity/attitude pairs, different regression models were tested to approximate the relationship the models (linear, quadratic and exponential). Since the maximum roll and pitch angle is 90° for a FWMAV, it is expected that the velocity will converge to a maximum value as the angle approaches 90°. Thus, the model is approximated as the exponential function in Eq. (6). Linear or quadratic models could be used as well, but they would only be valid within specific ranges since they do not converge to a constant value.

$$v(\theta) = a + be^{-c\theta} \tag{6}$$

Using the obtained steady-state velocity, a nonlinear regression model is obtained using Eq. (6). For both pitch and roll the parameters are initialised as: $a = -90, b = 90$ and $c = 0.3$. Moreover, to test the applicability of the exponential model, the same described procedure was applied to the data sets from [22] regarding the Delfly Nimble (a 33cm-wingspan FWMAV of the same type). Then the nonlinear regression model is applied initialising the parameters with exactly the same values. Figure 6 shows how the exponential curves of the Flapper and the Delfly properly approximate to the given data points for both roll and pitch.

6 Height control

Having a reliable height control strategy is essential to guarantee successful autonomous flight. Mainly it is used to keep the drone flying at a certain altitude and for hovering. Nevertheless, it also plays a major role in landing, which is known to be as the most challenging phase of flight for any aircraft. Height control is intimately related with thrust. Therefore, two approaches for improving thrust command were considered, first one is a voltage-dependent thrust model, and second one is an analysis of the ground effects.

6.1 *Voltage-thrust model*

Thrust is part of the position and velocity control of the drone, it also involves a PID controller which takes a commanded throttle and delivers an output signal for the motors. The signal from the controller is summed with a feed forward

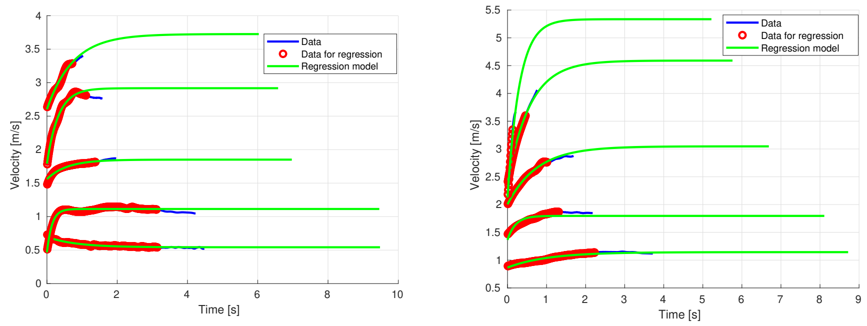


Figure 5: Extrapolation models for pitch (left) and roll (right) at different velocities

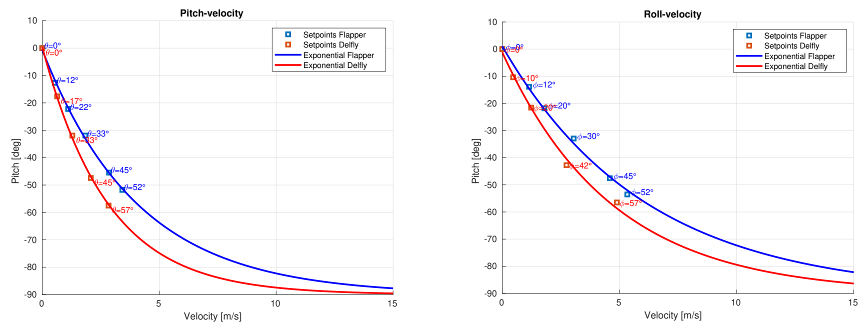


Figure 6: Pitch/velocity (left) and roll/velocity(right) models obtained from extrapolated data

term known as base thrust, which is the thrust required to keep the drone hovering at a certain altitude. In most cases the base thrust is a constant value, but there is evidence that the required base thrust tends to increase as the battery voltage drops off [13]. In order to observe this behaviour, a series of experiments were done where the drone started flying with a fully-charged battery and then let it hover until the battery got discharged. all values were sampled at 50 Hz and mapped as depicted in Figure 7.

Throttle is used instead of thrust because this is the signal that is directly input into the motors. In this case, throttle is a dimensionless value where 30000 sets the motors to the lowest speed and 60000 indicates full speed. The data points are then filtered and used to create a regression model. The improvement when increasing the order of the regression model is not significant. First, second and third-order models yield $R^2 \approx 0.756$. Albeit there is noisy data involved, it is not considered for the regression model since it is already known that the noise comes mainly from the feedback of the controller and the IMU measurements. Nevertheless, there is the option of using alternative regression models to account for the stochastic behaviour observed in Figure 7, which may result in a better R^2 value.

Once the model is implemented, the base thrust turns into a variable base thrust whose value will tend to increase the longer the drone keeps flying. To evaluate the model, a series of experiments was conducted using four different thrust controllers: a P and PI height controller, using both the constant and variable base thrust. For each controller, the flight was analysed in five different directions: X-motion, Y-motion, Z-motion, XZ-motion and YZ-motion. For each motion, a flight test was done consisting on five repetitions of step-input commands in the given motion, in order to get an average behaviour. Hence, Table 2 summarises the standard deviation and the mean absolute error between the transient response and the commanded height for each motion and controller. Notice from the table that for most of the motions, using a PI will result in lower error than a P controller, regardless of the

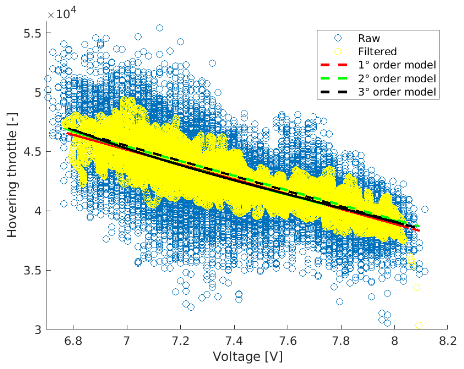


Figure 7: Throttle and voltage mapping with tendency lines for different regression models

variable or constant base thrust. For most conditions, variable PI reaches the lowest mean absolute error and standard deviation, showing the utility of varying thrust model.

Figure 8 depicts the averaged transient response in height for each controller, for the cases of X motion and XZ motion. The reduction of steady-state error caused by the integral gain is clear in the X motion, where the drone should keep flying at the same height when moving from one position to another. The XZ motion shows a case in which variable thrust model performs less well. In motions where a change in height occur, performance is more similar between P and PI controllers. From one side the integral gain tends to increase the settling time compared to the P controller, but at the same time the oscillations of the PI response reduces the error to minimum whenever it crosses the commanded value.

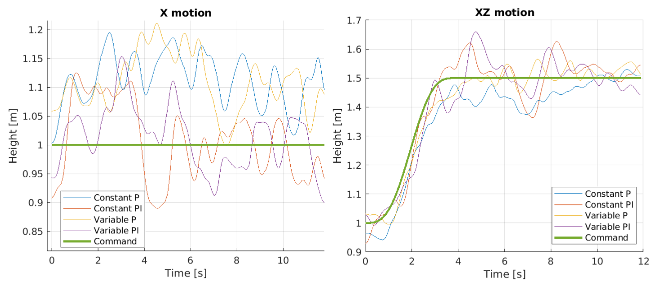


Figure 8: Extrapolation models for pitch (left) and roll (right) at different velocities

6.2 *Ground effect*

For the second approach in height control, the target was to investigate whether the ground effect has a large effect when flying low. According to [14], the range of height in which ground effects usually occur is $3d > h > d$, where d is the diameter of propellers in quadrotors. Similarly, for a FWMAV, the range would be proportional to the wingspan (0.5m). Hence, the experiments to model the ground effect consisted on keeping the drone hovering at different heights between 0.33m and 1.2m for over a minute. The lowest height tested was 0.33m above the ground, since this was the lowest height at which the drone can be kept hovering autonomously due to the location of the UWB anchors. In this section, the height is considered to be the distance between the landing gear and the ground. For each different height, a mean thrust and mean voltage is obtained. Using the thrust-voltage relationship found previously, the corrected thrust can be obtained. The ratio between the corrected thrust and the mean thrust is specific for each different height, as depicted in figure 9. Notice that for each data point an upper and lower bound is also provided based on the standard deviation of the thrust and battery voltage. According to [24], the relationship between thrust ratio and height can be modelled as:

$$\frac{T_{input}}{T_{output}} = 1 - \lambda \left(\frac{1}{h-a} \right)^2 \qquad (7)$$

After mapping the thrust ratio with their respective height, a non-linear regression model using equation Eq. (7) is applied. Such model approximates the coefficients $\lambda = 0.00093$ and $a = 0.4213$. In Figure 9, the regression model stays within the bounds of each data point. At the tested heights the ground effect still has relatively little influence on the thrust ratio. The thrust ratio only decreases to 92% at the lowest height of 0.33m. Thus, the ground effect is little up to 0.33m. Lower heights are not considered relevant, because, due to the disposition the UWB anchors, the drone is unlikely to fly lower except during take-off and landing. Nevertheless, the fit shows a sharp drop-off below 0.33m. Whether this is correct will have to be confirmed with future experiments.

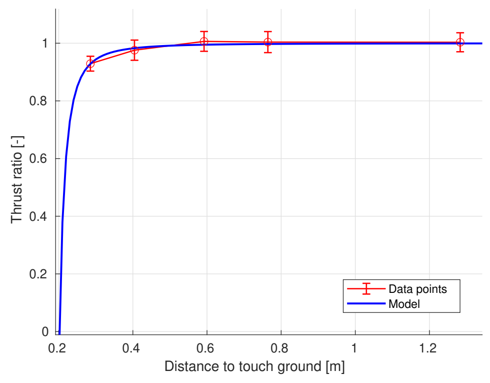


Figure 9: Thrust ratio for each height

.

7 Experimental validation

Once all the modifications were done, two versions of the controller were tested for validation. The first version is referred as "raw" version, which is the working implementation with the default EKF and PID controller mentioned in Section 3, without the proposed enhancements. The second version is referred as the "modified" version, and is the implementation with the adaptations mentioned in Sections 4, 5, and 6.

The validation consists of analysing the transient responses in position and velocity when position step-inputs were given in x, y, and z. For a wider perspective on the stability of the controller, the amplitude of the step input was changed from 1m to 2m. Thus, any difference in the aggressiveness of the response can be detected as well. For each motion and step-input, a series of five repetitions was done and averaged to obtain the general behaviour of the transient response. Table 3 presents how the mean absolute error, for both position and velocity, decreases when using the modified controller instead of the raw controller. In general, errors are approximately 1.5 times lower after the modifications.

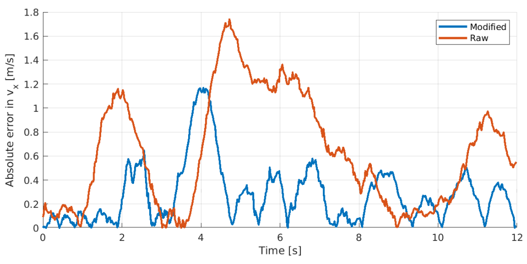
	X motion		Y motion		Z motion		XZ motion		YZ motion		Overall	
Controller	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
Constant P	0.109	0.08	0.098	**0.075**	0.077	0.117	**0.035**	0.166	0.043	0.143	0.072	0.116
Constant PI	0.061	0.159	0.048	0.093	**0.06**	0.075	0.047	**0.09**	0.047	0.1175	0.053	0.107
Variable P	0.114	**0.069**	0.086	0.079	0.065	0.069	0.057	0.116	0.046	0.1038	0.074	0.09
Variable PI	**0.049**	0.107	**0.025**	0.083	**0.06**	**0.056**	0.05	0.092	**0.037**	**0.061**	**0.044**	**0.08**

Table 2: Errors and standard deviations for each controller. Best result for each condition is bold-cased.

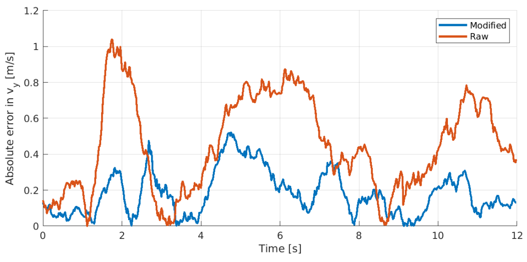
Mean absolute error for position step input 1						
	X motion		**Y motion**		**Z motion**	
Version	Raw	Modified	Raw	Modified	Raw	Modified
Position [m]	0.1706	**0.105**	0.1898	**0.1048**	0.0496	**0.043**
Velocity [m/s]	0.3945	**0.1297**	0.3001	**0.1199**	0.1001	**0.0892**
Mean absolute error for position step input 2						
	X motion		**Y motion**		**Z motion**	
Version	Raw	Modified	Raw	Modified	Raw	Modified
Position [m]	0.3316	**0.1978**	0.2989	**0.2055**	0.1245	**0.1097**
Velocity [m/s]	0.6330	**0.2859**	0.4583	**0.1802**	0.2476	**0.1613**

Table 3: Mean absolute errors of validation experiments. Best performance for each case is bold-cased.

Lastly, Figure 10 illustrates how the absolute error obtained in the transient response of the modified version is smaller and more consistent than the one of the raw version. For the sake of simplicity, only the transient responses of velocity in x and y to the step input of 2m are given, as the increase in performance in those two is the largest, according to Table 3. In Figure 10 the error is defined as the absolute difference between the output value and the corresponding commanded value. The plotted error corresponds to the transient response when the step input of amplitude 2 is given at $t = 1$s. Notice that almost throughout the whole response the error of the modified version is lower than the one of the raw version. Moreover, most of the peaks of the modified version are roughly 75% smaller than the ones from the raw version.



(a) Absolute error in Velocity X



(b) Absolute error in Velocity Y

Figure 10: Comparison of velocity error during transient response to a step input of amplitude 2

8 Conclusions

The achievement from this work was defining a strategy for enhancing position control for FWMAVs. From the estimation perspective, the strategy uses an extended Kalman filter to fuse UWB and IMU data, and it also incorporates a variable sensor noise term. Altogether, state estimation achieves accuracy between 8-10 cm error. From the control side, three specific aspects are considered.

Firstly, a velocity/attitude nonlinear model, which showed to be valid for two different kinds of FWMAVs. Thus, proving that it can be adapted, as long as there is experimental data from which the steady-state velocity can be extrapolated for a certain angle. Moreover, the model was validated using real flight data and proved its efficacy for drag compensation as part of the feed-forward term in the velocity control loop.

Secondly, the lower standard deviation of voltage-dependent thrust, compared to constant thrust, demonstrates a more consistent performance through time. A more notorious contrast can be obtained if the FWMAV flies for longer periods.

Thirdly, the ground effect experiments prove that as long as the drone's wings fly above 0.5m from the ground, no significant additional thrust will appear. Nevertheless, to know how much extra thrust is produced below 0.5m, experiments should be done through manual flight.

Lastly, in order to improve performance in autonomous flight using UWB, further development can be done in the position estimation. For instance drag is not considered in the EKF for the estimation of body velocities. An option would be deriving a drag model from IMU data. Other approach would be fusing data from other sources (e.g. barometer and magnetometer data) to compensate for the noisy accelerometer measurements due to inherent mechanical vibrations from flapping wing flight.

Acknowledgements

The realisation of this paper was possible thanks to the support of the start-up Flapper Drones B.V., and the MAVLab from the TU Delft Faculty of Aerospace.

REFERENCES

[1] Charles Porter Ellington. The aerodynamics of hovering insect flight. IV. Aerodynamic mechanisms. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 305(1122):79–113, 1984.

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

[3] Juan J Pomárico-Franquiz, Moises Granados-Cruz, and Yuriy S Shmaliy. Self-localization over RFID tag grid excess channels using extended filtering techniques. *IEEE Journal of Selected Topics in Signal Processing*, 9(2):229–238, 2014.

[4] Abdulrahman Alarifi, AbdulMalik Al-Salman, Mansour Alsaleh, Ahmad Alnafessah, Suheer Al-Hadhrami, Mai A Al-Ammar, and Hend S Al-Khalifa. Ultra wideband indoor positioning technologies: Analysis and recent advances. *Sensors*, 16(5):707, 2016.

[5] Bardia Alavi and Kaveh Pahlavan. Modeling of the TOA-based distance measurement error using UWB indoor radio measurements. *IEEE Communications Letters*, 10(4):275–277, 2006.

[6] Zengke Li, Guobin Chang, Jingxiang Gao, Jian Wang, and Alberto Hernandez. GPS/UWB/MEMS-IMU tightly coupled navigation with improved robust kalman filter. *Advances in Space Research*, 58(11):2424–2434, 2016.

[7] Qigao Fan, Yaheng Wu, Jing Hui, Lei Wu, Zhenzhong Yu, and Lijuan Zhou. Integrated navigation fusion strategy of INS/UWB for indoor carrier attitude angle and position synchronous tracking. *The Scientific World Journal*, 2014, 2014.

[8] Karl Martin Kajak, Matej Karásek, Qi Ping Chu, and GCHE De Croon. A minimal longitudinal dynamic model of a tailless flapping wing robot for control design. *Bioinspiration & Biomimetics*, 14(4):046008, 2019.

[9] Mostafa RA Nabawy and William J Crowther. The role of the leading edge vortex in lift augmentation of steadily revolving wings: a change in perspective. *Journal of the Royal Society Interface*, 14(132):20170159, 2017.

[10] Dong Xue, Bifeng Song, Wenping Song, Wenqing Yang, Wenfu Xu, and Tao Wu. Computational simulation and free flight validation of body vibration of flapping-wing mav in forward flight. *Aerospace Science and Technology*, 95:105491, 2019.

[11] JL Verboom, Sjoerd Tijmons, Christophe De Wagter, B Remes, Robert Babuska, and Guido CHE de Croon. Attitude and altitude estimation and control on board a flapping wing micro air vehicle. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5846–5851. IEEE, 2015.

[12] Zhi Ern Teoh, Sawyer B Fuller, Pakpong Chirarattananon, NO Prez-Arancibia, Jack D Greenberg, and Robert J Wood. A hovering flapping-wing microrobot with altitude control and passive upright stability. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3209–3216. IEEE, 2012.

[13] Diana A Olejnik, Bardienus P Duisterhof, Matej Karásek, Kirk YW Scheper, Tom Van Dijk, and Guido CHE De Croon. A tailless flapping wing mav performing monocular visual servoing tasks. *Unmanned Systems*, 8(04):287–294, 2020.

[14] IC Cheeseman and WE Bennett. The effect of the ground on a helicopter rotor. *R & M*, 3021, 1957.

[15] Sanjukta Aich, Chahat Ahuja, Tushar Gupta, and P Arulmozhivarman. Analysis of ground effect on multi-rotors. In *2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE)*, pages 236–241. IEEE, 2014.

[16] Yuan Xu, Yuriy S Shmaliy, Xiyuan Chen, Yueyang Li, and Wanfeng Ma. Robust inertial navigation system/ultra wide band integrated indoor quadrotor localization employing adaptive interacting multiple model-unbiased finite impulse response/kalman filter estimator. *Aerospace Science and Technology*, 98:105683, 2020.

[17] Steven van der Helm, Mario Coppola, Kimberly N McGuire, and Guido CHE de Croon. On-board range-based relative localization for micro air vehicles in indoor leader–follower flight. *Autonomous Robots*, 44(3):415–441, 2020.

[18] Matěj Karásek, Mustafa Percin, Torbjørn Cunis, Bas W van Oudheusden, Christophe De Wagter, Bart DW Remes, and Guido CHE de Croon. Accurate position control of a flapping-wing robot enabling free-flight flow visualisation in a wind tunnel. *International Journal of Micro Air Vehicles*, 11:1756829319833683, 2019.

[19] Matej Karasek. Flapper drones. https://flapper-drones.com/wp/. Accessed: 2021-01-01.

[20] Controllers in the crazyflie. https://www.bitcraze.io/documentation/repository/crazyflie-firmware/2020.04/functional-areas/controllers/. Accessed:2021-02-01.

[21] Mark W Mueller, Michael Hamer, and Raffaello D'Andrea. Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1730–1736, May 2015.

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

[23] Taimur Ali Shams, Syed Irtiza Ali Shah, Ali Javed, and Syed Hossein Raza Hamdani. Airfoil selection procedure, wind tunnel experimentation and implementation of 6dof modeling on a flying wing micro aerial vehicle. *Micromachines*, 11(6):553, 2020.

[24] 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. IEEE, 2015.

Onboard Time-Optimal Control for Tiny Quadcopters

Jelle Westenberger[1], Christophe De Wagter[1] and Guido C.H.E. de Croon[1] *

ABSTRACT

Time-optimal model predictive control is important for achieving fast racing drones but is computationally intensive and thereby rarely used onboard small quadcopters with limited computational resources. In this work, we simplify the optimal control problem (OCP) of the position loop for several maneuvers by exploiting the fact that the solution resembles a so-called 'bang-bang' in the critical direction, where only the switching time needs to be found. The non-critical direction uses a 'minimum effort' approach. The control parameters are obtained by means of bisection search schemes on an analytical path prediction model. The approach is compared with a classical PID controller and theoretical time-optimal trajectories in simulations. We explain the effects of the OCP simplifications and introduce a method of mitigating one of these effects. Finally, we have implemented the 'bang-bang' controller as a model predictive controller (MPC) onboard a Parrot Bebop and performed indoor flights to compare the controller's performance to a PID controller. We show that the light novel controller outperforms the PID controller in waypoint-to-waypoint flight while requiring only minimal knowledge of the quadcopter's dynamics.

1 INTRODUCTION

Unmanned air vehicles (UAV) are used in an increasing variety of applications [1]. Several applications, such as emergency response or race tasks require the drones to fly as fast as they can. Autonomous drone racing has recently emerged as a discipline to boost the development of fast-flying robots [2–4].

Traditionally the problem of time-optimal control generation is solved offboard as available hardware lacks the computational performance to quickly solve the Optimal Control Problem (OCP) onboard a quadcopter [5]. Fast flight is achieved by tracking these trajectories with high-performance controllers [6].

Recent work demonstrated efficient trajectory optimization for snap and leveraging differential flatness to derive the corresponding control inputs [7, 8]. However, the snap optimization method does not optimize for minimum-time. In fact, the total flight time must be predefined and the dynamical limits of the quadcopter are not taken into account. Including time and dynamic feasibility constraints in the optimization process increases the computational complexity of the problem [9]. On the other hand, [10] defines a sequential quadratic programming problem to simultaneously optimize control inputs for action and perception objectives. Albeit that in this work the reference trajectories are precomputed. [11] has extended on this work and demonstrated a pipeline that is fully embedded and is efficient enough to be implemented as a robust MPC. While the results are great, this comes at a very high computational cost. To address this, optimal control has also been approximated with deep neural nets, which are lighter than the original optimization [12–14]. This approach is powerful but very data intensive. Model predictive control remains very computationally expensive and few onboard implementations exist for very light drones [11, 15]. In this category, classical control remains common [16].

For a lot of trajectories, the time-optimal solution simplifies to a well-timed maximal control deflection. This paper therefore, proposes a light strategy to approximate time-optimal control by computing this timing onboard (See Figure 1).

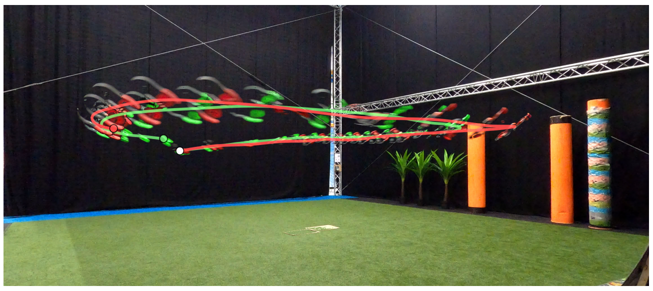


Figure 1: A comparison of a circular flight path between the proposed controller (green) and a classical PID controller (red)

Section 2 shows that the time-optimal position control simplifies to a bang-bang action on the attitude under well-selected conditions. In Section 3 we derive the differential equations that drive the proposed light MPC controller. Simulation results are presented in Section 4. Section 5 augments the model for the latency in attitude. Section 6 shows the re-

*[1]All authors are with the Faculty of Aerospace Engineering, Delft University of Technology, 2629 HS Delft, The Netherlands
jellewestenberger@gmail.com
c.dewagter@tudelft.nl
g.c.h.e.decroon@tudelft.nl

sults obtained onboard a Parrot Bebop before Section 7 gives the conclusions.

2 Simplified Time-Optimal Control

We have simplified the OCP by assuming a constant altitude and using the fact that for second-order systems the time-optimal solution consists of a 'bang-bang' motion. For quadcopter position control this translates to a double step in either pitch or roll with maximal amplitude. The OCP is hereby reduced to a problem in which the only parameter to-be optimized is the switching time, which reduces the computational complexity of the problem sufficiently to even allow implementation onboard very small quadcopters. The collective thrust is governed by the constant altitude assumption and is therefore considered to be always equal to $W/\cos\theta\cos\phi$, where W is the weight, θ and ϕ are the pitch and roll angles, respectively. The lateral dynamics can then be further simplified.

2.1 *Proving bang-bang solution for constant angles*

Hehn, Ritz, and D'Andrea [5] show in their work that with Pontryagin's minimum principle it can be proven that the time-optimal solution for a two-dimensional quadcopter trajectory consists of a bang-bang input in thrust and bang-singular-bang in rotational rate. We show that this solution remains valid in our simplified OCP in which we neglect the rotational dynamics, to further reduce the computational expense. Continuing with the Hamiltonian from [5]:

$$H(\mathbf{x},\mathbf{u},\mathbf{p}) = 1 + p_1\dot{x} + p_2 u_T \sin\theta + p_3\dot{z} + p_4(u_T\cos\theta - 1) + p_5 u_R \tag{1}$$

where u_T is the thrust input and u_R is the rotation rate input. p_i are the costates. Our model assumes instantaneous attitude changes and no changes in altitude. Therefore we can discard the last three terms of equation 1 and change the state θ to input u_θ. Pontryagin's minimum principle states that the optimal control input $\mathbf{u}^*$ minimizes the Hamiltonian [17].

$$\mathbf{u}^* = \operatorname{argmin} p_2 u_T \sin u_\theta \tag{2}$$

Depending on the sign of p_2, u_θ^* is either $\pm 0.5\pi$ or singular when $p_2 = 0$. However, at these pitch angles, it would be impossible to maintain altitude. Therefore, maximum pitch and roll angles are determined based on the thrust-to-weight performance of the quadcopter while reserving a margin of available thrust for additional control.

2.2 *Minimum-effort Approach*

We apply the 'bang-bang' solution to one critical axis. Intuitively, this axis is selected to be the direction with the largest initial position error. Control of the direction perpendicular to this axis is based on a 'minimum effort' approach. The intuition behind this approach is to only spend the minimum required thrust on decreasing the position error in the non-critical dimension such that a maximum available thrust can be spent on the critical dimension. This is achieved by calculating the constant attitude for which the non-critical position target is reached at the same time the critical target is reached.

3 Bang-Bang MPC

Based on the simplified OCP, we have created a controller that calculates the optimal roll and pitch angle from path predictions. We refer to this pipeline as the 'bang-bang' controller.

3.1 *Path Prediction*

For the sake of computational efficiency, we have simplified the dynamics such that the quadcopter's position and velocity can be evaluated analytically. By discarding the rotational and vertical dynamics, and partially decoupling the longitudinal and lateral dynamics we have derived a set 2nd order differential equations to describe the quadcopter's position and velocity.

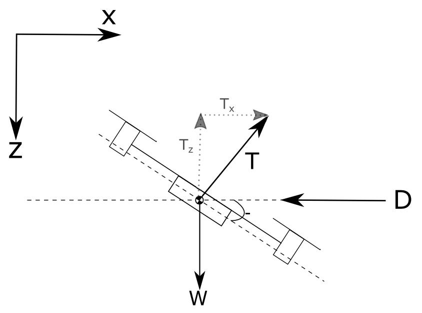


Figure 2: 2-D Quadcopter Dynamics

Based on the aforementioned assumptions and the force diagram depicted in Fig. 2 we state that the pitch angle θ and the thrust are constant and T_z equals the weight W. Furthermore, we assume that drag force D, consists only of flapping drag and is linearly proportional to airspeed $\dot{x}$, which is governed by drag coefficient C_d. So we can write:

$$\ddot{x} = g\tan\theta - \frac{C_d}{m}\dot{x} \tag{3}$$

Where g and m are the gravitational acceleration and mass, respectively.

Equation 3 is a 2nd order, non-homogeneous equation and is easily solved with the characteristic equation and method of undetermined coefficients. This yields:

$$x = c_1 e^{\frac{-C_d}{m}t} + c2 + \frac{W\tan\theta}{C_d}t \tag{4a}$$

$$\dot{x} = c_1\frac{-C_d}{m}e^{\frac{-C_d}{m}t} + \frac{W\tan\theta}{C_d} \tag{4b}$$

Constants c_1 and c_2 are solved with the quadcopter's initial position x_0 and initial velocity $\dot{x}_0$. This procedure can be repeated for the lateral direction, which is the direction out-of-plane in Fig. 2, taking into account the proper Euler angle rotations when deriving the lateral component of the thrust force:

$$y = c_3 e^{\frac{-C_d}{m}t} + c_4 + \frac{W}{\cos\theta}\frac{\tan\phi}{C_d}t \tag{5a}$$

$$\dot{y} = c_3 \frac{-C_d}{m} e^{\frac{-C_d}{m}t} + \frac{W}{\cos\theta}\frac{\tan\phi}{C_d} \tag{5b}$$

c_3 and c_4 are solved with the quadcopter's initial lateral position y_0 and lateral velocity $\dot{y}_0$.

Figure 3 illustrates an example of a single path prediction and the corresponding pitch and roll angles. The bang-bang maneuver of the longitudinal path can be described by evaluating equation 4 for two segments; One segment up to the switching instant (the acceleration phase) and a segment up to the time of arrival (the braking phase). Only the constants and pitch angles change between the two sets. The final velocity and position of the first segments are used as initial conditions for the second segment.
The lateral path can be described by one segment because in the 'minimum effort' approach the roll angle is assumed to be constant for the entire maneuver up to the target.

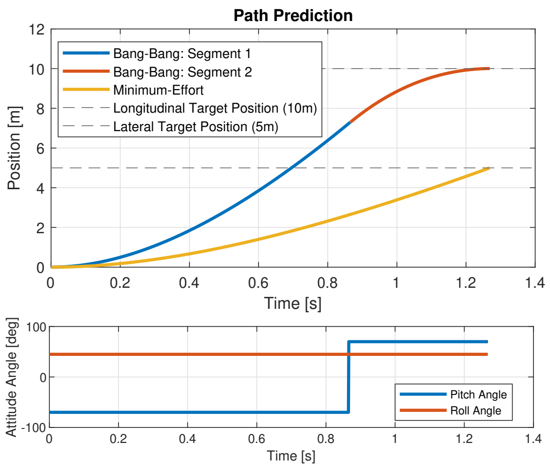


Figure 3: Example of predicted longitudinal and lateral paths. The longitudinal direction takes on a bang-bang motion that consists of two segments with opposing constant pitch angles. The lateral direction takes a minimum-effort approach which consists of a single segments and a constant roll angle.

3.2 *Optimizing*

In the OCP, the switching time and the non-critical angle are the two parameters that are to be optimized. Thanks to the analytical nature of the path equation a fast iterative bisection scheme can be used to find the optimal switching time and angle.

3.2.1 Solving Switching Time

To solve the switching time a desired velocity at the position target must be given in advance. The bisection scheme then iteratively adapts the switching time to minimize the velocity error at the target position. This procedure is described in Algorithm 1. In addition to an optimized switching time, an estimated time of arrival (ETA) is given as well. This is used in optimizing the non-critical angle.

Algorithm 1

```
$t_0 \leftarrow 0$
$t_1 \leftarrow$ initial guess
$E_t \leftarrow$ error threshold
$y_d \leftarrow$ desired position
while $E > E_T$ do
    $t_s \leftarrow \frac{t_0+t_1}{2}$
    $t_t \leftarrow get\_time\_from\_desired\_speed(v_d)$
    $E \leftarrow get\_position(t_t) - y_d$
    if $E > 0$ then
        $t_1 \leftarrow t_s$
    else
        $t_0 \leftarrow t_s$
    end if
end while
```

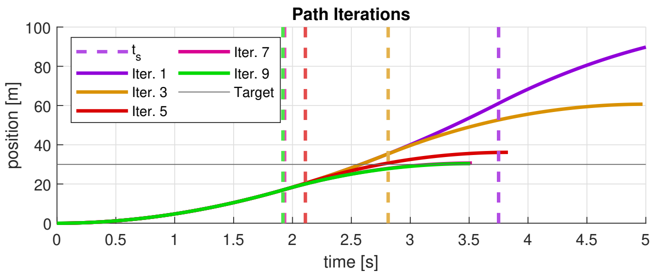


Figure 4: Illustration of the switching time, t_s, optimization process. The goal is to reach the target at 30m with zero rest speed.

3.2.2 Solving Minimum-Effort Angle

Analogously to the critical direction, a bisection scheme is used to iteratively change the angle to minimize the non-critical position error at the estimated time of arrival. The goal for the quadcopter is to reach the critical and non-critical targets simultaneously.
Moreover, during the braking phase of the 'bang-bang' motion, the related angle will also be optimized in this fashion to correct for prediction inaccuracies in this phase.

4 SIMULATIONS

Simulations have been performed to compare the 'bang-bang' controller flight performance to a classical PID controller and to time and snap optimized trajectories, provided

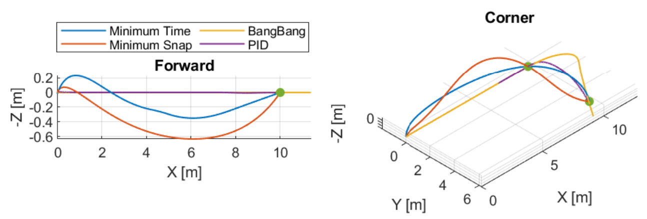


Figure 5: Trajectory comparison in simulation

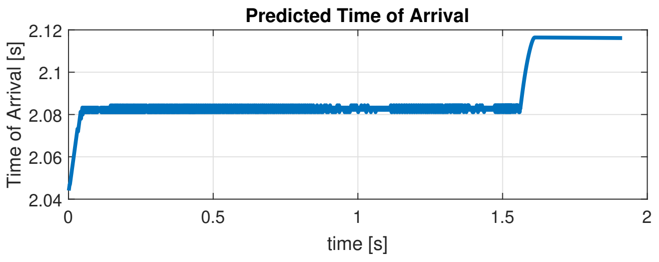


Figure 6: Development of the ETA of a simulated flight, corrected for passed simulation time. For perfect predictions, the time of arrival would be constant.

by the well-known ICLOCS toolbox [18]. This toolbox uses direct collocation to optimize a nonlinear OCP from an initial guess. Two maneuvers, a straight and a cornered trajectory, have been simulated to individually test the longitudinal and lateral flight behavior. The resulting flight times are summarized in table 1.

	Forward	Corner
Bang-Bang	1.52 s	2.53 s
PID	2.51 s	4.16 s
Min. Snap	1.77 s	2.53 s
Min. Time	1.30 s	1.87 s

Table 1: Simulated flight times

It can be seen that the bang-bang controller outperforms the PID controller in all maneuvers and is on par with the snap-optimized solution, but at a fraction of the computational cost.

5 TRANSITION COMPENSATION

It was found in simulations that the instantaneous angle assumption of the path predictor has the largest negative effect on the performance of the 'bang-bang' controller. Since a quadcopter cannot achieve infinitely high rotation rates the second part (further called the braking phase) of the bang-bang maneuver will always be initiated too late. As Figure 6 illustrates, path predictions deviate during the rotation from rest to acceleration at 0 s, and during the transition from accelerating to braking around 1.6 s.

In order to mitigate this issue, we have implemented a method that approximates the elapsed time and change of speed and position during the transition. Subsequently, the initial conditions of the braking phase are augmented with these values to improve the path predictions, as figure 7 illustrates. Figure 8 shows the effect of different degrees of compensation for a simulated flight.

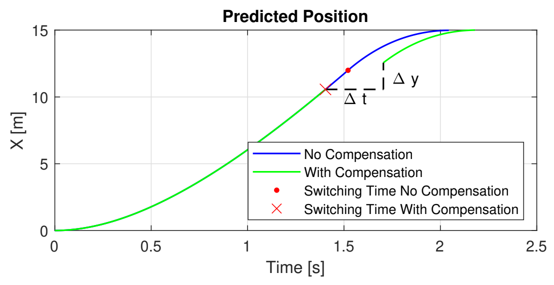


Figure 7: The initial conditions of the braking phase are shifted with $\Delta t, \Delta y$ and Δv to compensate for the rotation dynamics during the transition from accelerating to braking.

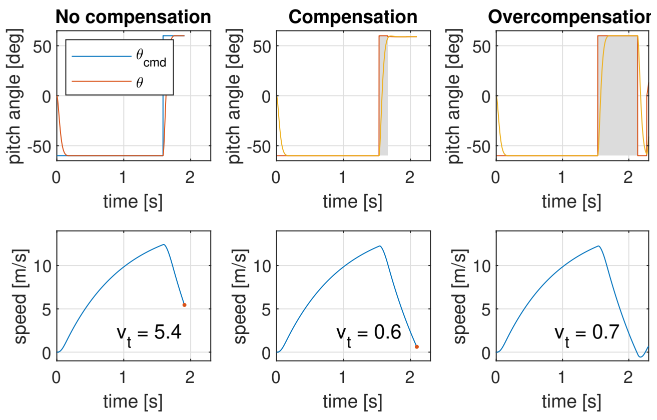


Figure 8: Different degrees of transition compensation are illustrated here. The shaded area indicates the period at which braking is forced. The desired target speed v_t is 0 m/s.

6 EXPERIMENTS

Flight tests have been performed with a commercial Parrot Bebop quadcopter in which all software was changed. The performance of the 'bang-bang' MPC is compared to a traditional PID controller for different types of maneuvers.

6.1 *Experimental Setup*

The 'bang-bang' controller has been implemented in the open-source autopilot framework Paparazzi-UAV [19] and is executed onboard a Parrot Bebop quadcopter. The flights were performed in TU Delft's 'CyberZoo' indoor flight area outfitted with an Optitrack position and attitude tracking system. The position and heading are sent to the drone via WiFi and the state estimation is executed on board by means of the inertial measurement unit and complementary filters. The ‘bang-bang’ MPC and PID controller give roll and pitch commands while the inner control loop, based on Incremental Nonlinear Dynamic Inversion (INDI) [20], controls the rotational rates. Figure 9 gives an overview of the control pipeline.

6.1.1 PID Controller

The PID controller is a high-gain cascaded position-velocity controller. That is, the position error will govern the desired speed, which in turn governs the pitch and roll commands. A single set of gain values has been selected that gives the best overall result in all tests. This set is kept constant throughout all flight experiments. Furthermore, we have defined saturation limits for the allowable speed and pitch and roll angles. For a fair comparison, the same limits have also been applied to the ‘bang-bang’ controller.

6.2 Transition Estimators

As discussed in section 5, we can compensate for the unmodeled transition dynamics by approximating the transition losses. These dynamics are difficult and costly to simulate for a real quadcopter, therefore we have derived a simple linear regression model to approximate transition losses $\Delta t, \Delta y$ and Δv from flight measurements. We assumed that these losses are a function of the speed at the switching time, and of the total angle the quadcopter needs to rotate. We found that the transition losses varied between forward, backward and sideways flight maneuvers. Therefore, three different sets of estimators have been derived, each corresponding with one of these directions. In the control pipeline, one set of estimators is selected based on the direction in which a ‘bang-bang’ maneuver is planned.

6.3 Motion Primitives Flights

Test flights have been performed to test the bang-bang controller for different motion primitives. That is, six different maneuvers have been established to test the longitudinal and lateral performance in which the quadcopter starts and ends at rest. For each maneuver, a comparison is made between the ‘bang-bang’ controller, the ‘bang-bang’ controller with transition compensation, and the high-gain PID controller.
During these maneuvers the heading is kept constant and the critical direction for which the ’bang-bang’ maneuver is planned is based on the largest component of the initial position error.
Table 2 lists all maneuvers and their initial and target positions. The controllers are assessed on the time it takes to reach their target, the degree of overshoot, and the velocity error while passing the target.

Maneuver	Initial $\rightarrow$ Target Position (x,y,z) [m]
Forward	$(-2,0,1.5) \rightarrow (2.5,0,1.5)$
Backward	$(-2,0,1.5) \rightarrow (2.5,0,1.5)$
Sideways	$(0,-2,1.5) \rightarrow (0,2,1.5)$
Forward-Sideways	$(-2,-2,1.5) \rightarrow (2,1,1.5)$
Forward-Up	$(-2,0,1) \rightarrow (2.5,0,2.75)$
Forward-Down	$(-2,0,2.75) \rightarrow (2.5,0,1)$

Table 2: Translated distances for each motion primitive maneuver.

6.3.1 Results

Table 3 summarizes the flight results. It can be seen that the non-compensated bang-bang controller has the largest velocity errors and overshoot. Furthermore, it becomes obvious that the compensation system has a positive effect on the path prediction performance. Unfortunately, the transition loss model is not accurate enough to completely mitigate the transition losses and some degree of overshoot still occurs. In these simple start-stop tests, the PID controller is marginally slower than both bang-bang controllers but has lower overshoot and velocity errors.

6.4 Consecutive Waypoints Flight

To test the proposed controller in a setting that more closely resembles an autonomous drone race, a flight plan with consecutive positional waypoints has been implemented. In this flight plan, the quadcopter is no longer instructed to come to a full stop at each waypoint. The desired speed at each waypoint has been set to 2 m/s as it was found iteratively that this value in combination with a position threshold of 70 cm resulted in smooth trajectories for both the PID and bang-bang controllers (See Figure 1). However, it is expected that the optimal threshold values are controller- and trajectory-dependent.
Because currently no heading changes were incorporated into the ‘bang-bang’ maneuver planning, the heading is kept constant. The critical direction in which a ‘bang-bang’ motion is planned is automatically adjusted based on the direction with the largest position error.

6.4.1 Results

Figure 10 shows top views of flights with the two controllers. Both controllers have been assessed on the time it takes to complete one circle and the minimum position error. The results are displayed in figure 11. Here, the ‘bang-bang’ controller is seen to outperform the PID both in speed and target accuracy. The PID controller is unable to give priority to one direction over the other. Due to the high gain values, roll and pitch angles are quickly saturated even if the position error of one direction is much smaller than the other, which slows down the critical axis. In the various flight runs of the PID controller, large lateral oscillations can be seen. It was also

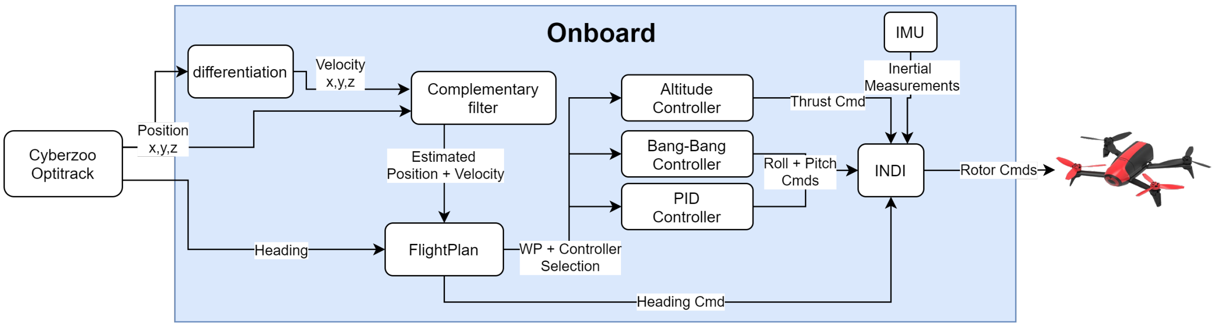


Figure 9: Control Pipeline

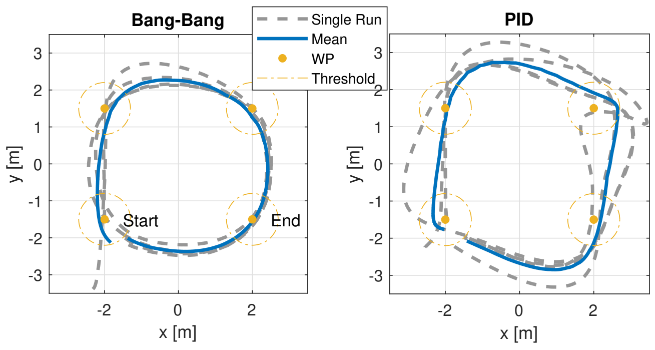


Figure 10: Top-view of consecutive waypoints flights

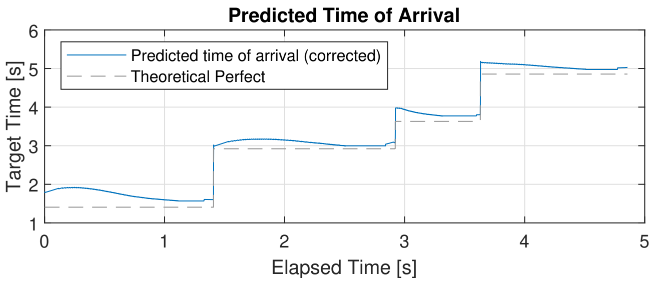


Figure 12: Estimated time of arrival corrected for passed time.

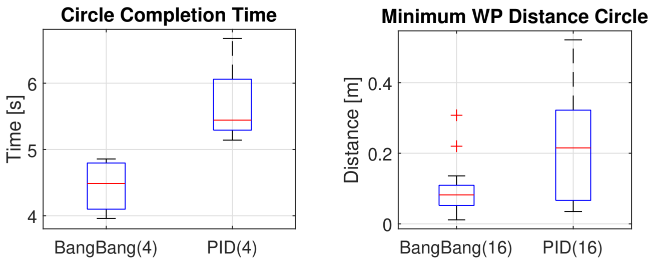


Figure 11: Circle completion time and minimal waypoint distance for consecutive waypoint flights

found that the PID controller was more likely to reach unstable situations due to the high simultaneous pitch and roll angles compared to the bang-bang controller.

The predicted times of arrival for a single run are illustrated in Figure 12. From this figure, we can derive the real-time path prediction performance. As expected the time increases during the angular rotations. However, during the acceleration phases, the time is seen to decrease. We think that this is caused by inaccurate aerodynamic drag estimation and by the effect the non-critical angle has on the acceleration in the critical direction.

7 CONCLUSIONS

We have proposed the 'bang-bang' MPC which approaches time-optimal control principles while being computationally efficient enough to run onboard a commercial quadcopter. This is achieved by assuming that the solution consists of a double step control input in attitude angle for one 'critical' direction while the non-critical direction has a constant angle as solution. This simplifies the OCP and drastically reduces the computational complexity. For efficient control parameter optimization, a bisection scheme in combination with an analytical path prediction model is used. We have shown both in simulation and in real-world flights that the 'bang-bang' controller is a feasible option for fast flight. In fact, for the consecutive waypoint flight, the 'bang-bang' controller is shown to be 17.5% faster than a traditional high-gain PID controller on average. The entire control pipeline easily runs at the main control loop frequency of 512Hz on the Bebop and is sufficiently light to run on even smaller and computationally-limited quadcopters. However, a more thorough analysis is needed to quantify the required computational effort.

The bang-bang MPC also shows promise to be an attractive easy-to-implement solution. For the pipeline requires minimal knowledge of the dynamics (only C_d and mass). And despite that the transition compensation in its current state relies on measurement data, future work could mitigate this process with on-line transition loss estimation. Currently, the constant altitude constraint forms the largest deviation from the theoretical time-optimum solution found by ICLOCS. Finally, the pitch, roll, and thrust limits are set conservatively

Controller	Maneuver					
	Forward	Backward	Sideways	Forward-Sideways	Forward-Up	Forward-Down
	Mean Time of Arrival [s]					
Bang-Bang	**1.38** (n=4)	**1.41** (n=4)	**1.29** (n=5)	**1.47** (n=4)	**1.25** (n=3)	**1.50** (n=3)
Bang-Bang Comp.	1.42 (n=15)	1.47 (n=12)	1.37 (n=11)	1.53 (n=15)	1.32 (n=7)	1.52 (n=7)
PID	1.48 (n=10)	1.54 (n=8)	1.43 (n=8)	1.51 (n=8)	1.40 (n=5)	1.54 (n=5)
	Mean Overshoot [m]					
Bang-Bang	0.62 (n=4)	0.81 (n=4)	0.53 (n=5)	0.27 (n=4)	1.20 (n=3)	0.77 (n=3)
Bang-Bang Comp.	0.18 (n=15)	0.22 (n=12)	0.06 (n=11)	0.11 (n=15)	0.51 (n=7)	0.20 (n=7)
PID	**0.05** (n=10)	**0.04** (n=8)	**0.04** (n=8)	**0.04** (n=8)	**0.03** (n=5)	**0.14** (n=5)
	Mean Velocity Error [$\frac{m}{s}$]					
Bang-Bang	3.05 (n=4)	3.51 (n=4)	3.06 (n=5)	1.85 (n=4)	3.82 (n=3)	3.38 (n=3).
Bang-Bang Comp.	1.60 (n=15)	1.88 (n=12)	0.69 (n=11)	0.38 (n=15)	2.63 (n=7)	1.63 (n=7)
PID	**0.08** (n=10)	**0.08** (n=8)	**0.14** (n=8)	**0.06** (n=8)	**1.01** (n=5)	**0.20** (n=5)

Table 3: Performance values the different controllers in 6 maneuvers. *n* is the number of runs performed.

and flight performance could be improved if these parameters are made adaptive.

REFERENCES

[1] M. Hassanalian and A. Abdelkefi. Classifications, applications, and design challenges of drones: A review. *Progress in Aerospace Sciences*, 91:99–131, May 2017.

[2] Hyungpil Moon, Yu Sun, Jacky Baltes, and Si Jung Kim. The IROS 2016 Competitions [Competitions]. *IEEE Robotics and Automation Magazine*, 24(1):20–29, 2017.

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

[4] Philipp Foehn, Dario Brescianini, Elia Kaufmann, Titus Cieslewski, Mathias Gehrig, Manasi Muglikar, and Davide Scaramuzza. AlphaPilot: Autonomous Drone Racing. In *RSS 2020*, 2020.

[5] Markus Hehn, Robin Ritz, and Raffaello D'Andrea. Performance benchmarking of quadrotor systems using time-optimal control. *Autonomous Robots*, 33(1-2):69–88, 2012.

[6] Ezra Tal and Sertac Karaman. Accurate Tracking of Aggressive Quadrotor Trajectories Using Incremental Nonlinear Dynamic Inversion and Differential Flatness. *Proceedings of the IEEE Conference on Decision and Control*, 2018-Decem:4282–4288, 2019.

[7] 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. IEEE, 2011.

[8] Matthias Faessler, Antonio Franchi, and Davide Scaramuzza. Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories. *IEEE Robotics and Automation Letters*, 3(2):620–626, 12 2017.

[9] Charles Richter, Adam Bry, and Nicholas Roy. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. *Springer Tracts in Advanced Robotics*, 114(Isrr):649–666, 2016.

[10] Davide Falanga, Philipp Foehn, Peng Lu, and Davide Scaramuzza. PAMPC: Perception-aware model predictive control for quadrotors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, October 2018.

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

[12] Antonio Loquercio, Elia Kaufmann, Rene Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scara-

muzza. Deep Drone Racing: From Simulation to Reality With Domain Randomization. *IEEE Transactions on Robotics*, pages 1–14, 2019.

[13] Shuo Li, Ekin Öztürk, Christophe De Wagter, Guido C. H. E. de Croon, and Dario Izzo. Aggressive online control of a quadrotor via deep network representations of optimality principles. In *2020 IEEE International Conference on Robotics and Automation, ICRA 2020*, pages 6282–6287, United States, May 2020. IEEE.

[14] Elia Kaufmann, Antonio Loquercio, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Deep Drone Acrobatics. *RSS: Robotics, Science, and Systems*, 2020.

[15] Tomas Baca, Giuseppe Loianno, and Martin Saska. Embedded model predictive control of unmanned micro aerial vehicles. In *2016 21st International Conference on Methods and Models in Automation and Robotics (MMAR)*. IEEE, 8 2016.

[16] Shuo Li, Erik Horst, Philipp Duernay, Christophe De Wagter, and Guido C. H. E. Croon. Visual model-predictive localization for computationally efficient autonomous racing of a 72-gram drone. *Journal of Field Robotics*, 37(4):667–692, 5 2020.

[17] Donald E Kirk. *Optimal control theory: An introduction*, volume 1. Dover Publications, 1998.

[18] Yuanbo Nie, Omar Faqir, and Eric C. Kerrigan. ICLOCS2: Try this Optimal Control Problem Solver before you Try the Rest. *2018 UKACC 12th International Conference on Control, CONTROL 2018*, 2(2017):336, 2018.

[19] Pascal Brisset, Antoine Drouin, Michel Gorraz, Pierre-Selim Huard, and Jeremy Tyler. The paparazzi solution. In *MAV 2006, 2nd US-European competition and workshop on micro air vehicles*. Citeseer, 2006.

[20] Ewoud J.J. Smeur, Qiping Chu, and Guido C.H.E. De Croon. Adaptive incremental nonlinear dynamic inversion for attitude control of micro air vehicles. *Journal of Guidance, Control, and Dynamics*, 39(3):450–461, 2016.

Fast simulation model for control law design and benchmark of high aspect ratio flexible UAVs

Romain. Jan (ISAE-SUPAERO, University of Toulouse, 31400 Toulouse, France)[*], Jean-Philippe Condomines (ENAC,31400 Toulouse, France),Jean-Marc Moschetta(ISAE-SUPAERO)

ABSTRACT

A low/middle range fidelity model is highlighted in this work through the development of a conservative extension in order to assess a significant part of the UAVs community's current issues. Such as flutter control, gust alleviation, trajectory tracking, energy harvesting strategies, parametric studies... One aims at providing an efficient tool for the quick design of high aspect ratio UAVs coupled with advanced control laws. The work recalls the theoretical background used on the model, the methodology applied to enhanced it, and 3 illustrative examples.

1 INTRODUCTION

Nowadays' the trend regarding UAVs is the improvement of efficiency, endurance, and impact on the environment leading to lighter and flexible aircraft, rising new problems regarding its sensitivity to the surrounding environment.

The last works regarding UAVs rise the need for modular and fast simulation model to capture sophisticated flight dynamics as well as the structure behaviours both coupled with advanced control strategies. Such a tool could assess the different issues risen by UAVs such as flutter damping ([1], [2],[3]), energy harvesting strategies ([4], [5], [6]), flap inversion behaviour due to structural deformation ([7]), flight envelopppee extension ([8]), gust alleviation ([9],[10])

There exists differents tools to asses those problems adapted to high aspect ratio UAVs such as ASWING [11], UM/NAST [12], CA^2LM [13] and others used in [14]. Despite the recency of some of them, after an overview of the theoretical background used, the authors choice converged to ASWING for 3 main reasons derived below:

structural part:
Some of them do not consider all the 6 degrees of freedom, inertial coupling, and local damping. Moreover, coupling between local beam variables is not considered (cross-sectional bending/twisting). Few models assume small beam deflections and a linear beam behavior. Elastic, tension, and gravity's center's positions can not be specified. Others use a rectangular beam assumption for bending and stiffness matrices computation.

aerodynamic part:

[*]Email address(es): romain.jan@isae-supaero.fr

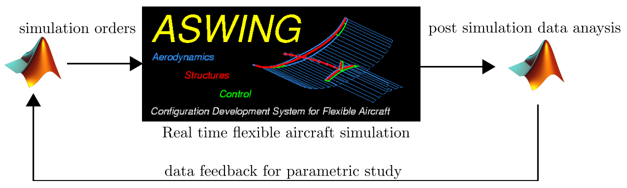

simulation orders
ASWING
Aerodynamics
Structures
Control
Configuration Development System for Flexible Aircraft
post simulation data anaysis
Real time flexible aircraft simulation
data feedback for parametric study

Figure 1: ASWING working with MATLAB

Propeller jet-induced velocities are neglected in most of cases. For some compressibility effects are not taken into account as well as ground effects. Shed vortices are approximated by a temporal Theodorsen function. Other models do not present a post-stall behavior capture.

applied forces:
Added mass effects, unsteady effects on propeller blades, joints, and struts loads are not usually considered as well. Joints degrees of freedom restriction, and concentrated efforts neither.

Furthermore, as most of those low order models are designed on MATLAB, calling external libraries costs a lot of time especially in temporal loops. Memory is dynamically allocated and freed leading to slower simulation. As a compiled program ASWING is very fast and works in real-time with nowadays computers. With an improved temporal solver and a compiled version on a dedicated device, it could show interesting performances.

However, MATLAB remains a very efficient and productive tool for analysis and post-simulation data treatment. Consequently, one's objective is to present an extension of ASWING that the user could run through MATLAB as an "opened" black box as shown in figure 1 The use of a fast compiled simulation software with MATLAB would allow large parametric studies of flexible UAVs for a small computational time cost.

Unfortunately, ASWING provides a very limited control law toolbox which limits its use in the flight control community highlighting the main contribution of this work.

2 PROBLEM STATEMENT

In ASWING [15],[16] the dynamic of a given aircraft is described by a non linear system with x as a states space vector:

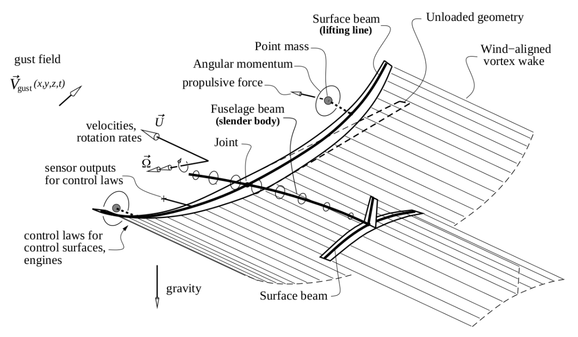


Figure 2: ASWING configuration representation [15]

$x =$

$(\vec{r_i}, \theta_i, M_i, u_i, \omega_i, \dot{r_i}, \dot{\omega_i}$: **beam node variables** $(18N_B)$

r_J, θ_J, M_J, F_J; **joints variables** $(12N_J)$

$A_1, \ldots, A_K, \dot{A_1}, \ldots, \dot{A_K}$: **circulation modes (2K)**

$\vec{R_E}, \dot{\vec{R_E}}, \vec{\Theta}, \dot{\vec{\Theta}}, \vec{U}, \dot{\vec{U}}, \vec{\Omega}, \dot{\vec{\Omega}}$: **global position and rates (32)**

$\vec{a_0}, \vec{\alpha_0}$: **absolute accelerations variables (6)**

$\delta F_1, \ldots \delta F_N, \delta E_1, \delta E_N$: **flaps and engine level (32)**

$)^T$

y is the output variables measured at each sensor location which are x dependant. The feedback-able variables are the local freestream velocity, angle of attack and sideslip, local positions, angles, velocities, rotations rates and accelerations, internal beam loads and circulations.
u denotes the forcing vector defined by :

$u =$

$(y^*$: **desired output behaviour**

$\delta^* F_1, \ldots \delta^* F_N, \delta^* E_1, \delta^* E_N$: **desired flaps, engine levels**

$V_{wind}(\vec{x}, z, y, t)$: **turbulence velocity model**

$)^T$

The 6 degrees of freedom of every beam node are governed by the non-linear Euler Bernouilli beam equations in its multivariable differential form. By denoting $q = (\vec{r_i}, \vec{\theta_i})_i \in [1, N_B]$.

$$[M(x)]\ddot{q} + [C(x)]\dot{q} + [K(x)]q = 0 \tag{1}$$

where $[M(x)]$, $[C(x)]$ and $[K(x)]$ are non linear mass, damping and stiffness matrices. They take into account the effect of structural damping, loads and rates, external distributed and concentrated loads. Distributed loads are lift and drag, added mass effect, inertial and gravity loads integrated along the beam. Concentrated loads recover the effect of engine thrust and drag, point mass, struts, and joints loads.
Joints variables are introduced to connect and constraint beams. Circulation modes are used in an extended unsteady lifting line theory taking into account compressibility effect and velocity influence of wing thickness; shed and steady vortices of every lifting beam and propeller jet stream. Circulation modes are constraints in a flow tangency condition with post-stall modelization. Those circulation modes are used to compute lift and drag. Distributed lift is computed with the unsteady form of Kutta-Joukowsky theorem for surface beams, and with slender body theory for fuselage beams. Distributed drag sums up the friction and pressure drag and post-stall contributions. For both lift and drag, the knowledge of tangent and orthogonal relative stream velocity is necessary which is a function of infinite freestream, local beam node positions and rates, turbulence, and induced velocities.
The global position, rates, and absolute accelerations variables are governed by the "rigid" kinematic and trajectory equations known as :

$$\frac{d\vec{R}_E}{dt} - \bar{\bar{T}}_E \vec{U} = 0 \tag{2}$$

$$\frac{d\vec{\Theta}}{dt} - \bar{\bar{C}}_E \vec{\Omega} = 0 \tag{3}$$

$$\frac{d\vec{U}}{dt} + \vec{\Omega} \times \vec{U} - \vec{a}_o = 0 \tag{4}$$

$$\frac{d\vec{\Omega}}{dt} - \vec{\alpha}_o = 0 \tag{5}$$

where $\bar{T}_E$ and $\bar{C}_E$ are global position and orientation dependant transformation matrices. Moreover, absolute accelerations can be constrained for a free flight or anchored configuration.

The flaps and engine-level variables are governed by a control law equation introduced later in the document.

In the end, the behavior of the state space vector x is governed by the non linear system :

$$(\Sigma) : \begin{cases} \dot{x} = f(x, u) \\ y = h(x, u) \end{cases} \tag{6}$$

Depending on the number of beams and their associated mesh, the total number of states variables can rise very fast and reach more than 10000. Using this discrete model as an analytical tool for control law design would be unadapted. Therefore ASWING provides a reduced-order model (ROM) generator function leading to the linear system

$$(ROM) : \begin{cases} \dot{z} = \Lambda z + Bu \\ y = Cz + Du \end{cases} \tag{7}$$

where Λ is a diagonal matrix containing the aircraft modes and z is an alternative state vector equivalent to x defined as:

$$z = Vx \tag{8}$$

Control laws	
Linear	**Non Linear**
PID	Gain Scheduling
LQG	Slidding mode control
H_∞/H_2	Control-Lyapunov functions
μ-analysis	Backstepping
Guardian mapping	Non linear damping
Fractional order Controller	Feedback Linearization

Table 1: Examples of implementable control laws

In practice, the user can select slower and fast undamped or unstable modes to drastically reduced the dimension of z. Moreover, he can select different outputs to study the small perturbations' effects of each mode. The system (7) is then a useful intel for control laws design.

Unfortunately, ASWING comes with a limited control law known as outputs proportional feedback :

$$\delta = \delta_{ref} + K_y(y - y_{ref}) \tag{9}$$

where K_y is a bi-scheduled matrix. One must note that integral errors are part of the y vector allowing the user to implement PID controllers. However the main problems of this form are the assumptions of perfect knowledge of the outputs and direct impact of actuators (no noise, no sensors/actuators dynamics or saturations), and the lack of the controller's internal dynamic. Consequently, advanced control synthesis and benchmark are limited leading to the contribution of this article.

3 ASWING ADVANCED CONTROL LAWS

One is seeking to implement a multi-inputs multi-outputs (MIMO) non-linear control law as depicted on figure 3. To fulfill this objective, one must introduce a new state vector to consider its internal dynamic. Thus for a set of control states variables vector x_c the control law takes the form:

$$(C) : \begin{cases} \dot{x_c} = f_c(x_c, y, y_{ref}, \delta_{ref}) = f(x_c, u_c) \\ \delta = h_c(x_c, u_c) \end{cases} \tag{10}$$

10 recovers many control law forms used in MIMO non linear and linear theory sum up in table 1. Moreover, this form remains conservative, meaning that the original PID can still be implemented. Thus the next lines aim at giving the critical steps to apply so that such control laws could be used on ASWING.

3.1 Modal analysis and time marching

For the next lines one will prefer the residual form of 6 written as :

$$r(x, \dot{x}, u) = \dot{x} - f(x, u) \tag{11}$$

The state vector temporal evolution can be solved by using a multivariable Newton alogorithm depicted in [17] and [18] as :

$$\mathrm{x}_n^{i+1} = \mathrm{x}_n^i - \left[\frac{\partial r}{\partial x} + k_0 \frac{\partial r}{\partial \dot{x}}\right]_i^{-1} r^i \tag{12}$$

where i is the Newton iteration index and n is the time index such that $t = nTe$. Again for the next lines one will prefer the shorter notation of the matrix:

$$\left[\frac{\partial r}{\partial x, \dot{x}}\right]_i = \left[\frac{\partial r}{\partial x} + k_0 \frac{\partial \mathbf{r}}{\partial \dot{x}}\right]^i \tag{13}$$

To solve 12 a Gaussian block elimination [17] is used to invert the matrix (13). Such matrix representation appears suitable for different types of analysis. In fact by forcing k_0 to zero, one force 12 to the steady case and by inspection of 13 eigenvalue one recovers the modal response of the aircraft for a given steady flight condition. Secondly, it provides an accurate time-marching behavior for a relatively small time step choice. To embed 10 in the solver 12 one must split the state vector x such as :

$$x^T = \left(x_Q^T, x_V^T, x_P^T, x_D^T\right) \tag{14}$$

with:

$$\begin{aligned}
x_Q^T &= \left(\vec{r}_i \quad \vec{\theta}_i \quad \vec{M}_i \quad \vec{F}_i \quad \vec{u}_i \quad \vec{\omega}_i\right)_{i\in[1,N_B]} \\
x_V^T &= \Big(\Delta\vec{r}_J \quad \Delta\vec{\theta}_J \quad \vec{M}_J \quad \vec{F}_J \quad A_1 \quad A_2 \\
&\quad \ldots \quad A_K \quad e\Big)_{J\in[1,N_J]} \\
x_P^T &= \left(\vec{R}_E \quad \vec{\Theta} \quad \vec{U} \quad \vec{\Omega} \quad \vec{a}_o \quad \vec{\alpha}_o\right) \\
x_D^T &= (\delta_F 1, \ldots, \delta_F N, \delta_E 1, \ldots, \delta_E N)^T
\end{aligned}$$

where evey variables from x_Q, x_V, x_P, x_D have been described in section 2. At the light of the state vector form, 12 is equivalent to invert the matrix:

$$\left[\frac{\partial r}{\partial x, \dot{x}}\right]_i^{-1} =
\begin{bmatrix}
\left[\frac{\partial r_Q}{\partial Q, \dot{Q}}\right]_i & \left[\frac{\partial r_Q}{\partial V, \dot{V}}\right]_i & \left[\frac{\partial r_Q}{\partial P, \dot{P}}\right]_i & \left[\frac{\partial r_Q}{\partial D, \dot{D}}\right]_i \\
\left[\frac{\partial r_V}{\partial Q, \dot{Q}}\right]_i & \left[\frac{\partial r_V}{\partial V, \dot{V}}\right]_i & \left[\frac{\partial r_V}{\partial P, \dot{P}}\right]_i & \left[\frac{\partial r_V}{\partial D, \dot{D}}\right]_i \\
\left[\frac{\partial r_P}{\partial Q, \dot{Q}}\right]_i & \left[\frac{\partial r_P}{\partial V, \dot{V}}\right]_i & \left[\frac{\partial r_P}{\partial P, \dot{P}}\right]_i & \left[\frac{\partial r_P}{\partial D, \dot{D}}\right]_i \\
\left[\frac{\partial r_D}{\partial Q, \dot{Q}}\right]_i & \left[\frac{\partial r_D}{\partial V, \dot{V}}\right]_i & \left[\frac{\partial r_D}{\partial P, \dot{P}}\right]_i & \left[\frac{\partial r_D}{\partial D, \dot{D}}\right]_i
\end{bmatrix}^{-1} \tag{15}$$

As mentioned before a Gaussian block elimination of order 4 is used to solve 15. Note that, the jacobian matrix $\left[\frac{\partial r_Q}{\partial Q, \dot{Q}}\right]_i$ follows a bi-tridiagonal block pattern. Thus the order of definition of the state vector in Eq. (14) is relevant because the first step of Gaussian Block elimination will not affect the upper left matrix block. For the rest of the inversion, one uses

LU factorization to invert the diagonal matrix blocks. Consequently one must not change the order of definition of the state vector, otherwise, a huge slow down effect during simulation will be witnessed.

3.2 *Extension to a control law with an internal dynamic*

As one rises the state vector x with x_c, the problem becomes:

$$\begin{bmatrix} \left[\frac{\partial r}{\partial x,\dot{x}}\right]_i & \left[\frac{\partial r}{\partial x_c,\dot{x_c}}\right]_i & \vdots & R_x \\ \left[\frac{\partial r_c}{\partial x,\dot{x}}\right]_i & \left[\frac{\partial r_c}{\partial x_c,\dot{x_c}}\right]_i & \vdots & Rx_c \end{bmatrix} \tag{16}$$

which leads again to the seek of:

$$\left[\frac{\partial r}{\partial x,\dot{x}}\right]_i^{-1} = \begin{bmatrix} \left[\frac{\partial r}{\partial x,\dot{x}}\right]_i & \left[\frac{\partial r}{\partial x_c,\dot{x_c}}\right]_i \\ \left[\frac{\partial r_c}{\partial x,\dot{x}}\right]_i & \left[\frac{\partial r_c}{\partial x_c,\dot{x_c}}\right]_i \end{bmatrix}^{-1} \tag{17}$$

As one was seeking to modify as less as possible ASWING code and as it implements a 4th order Gaussian Block Elimination and not a generic one, it is consequently not possible to just add the Jacobian matrix associated to X_c and R_{X_c} (bottom right block in (17)). Consequently, the code has been modified so that the new set of control law x_c is a part of the x_D state vector.

Note that it's not necessary to give all the jacobian matrices. Concidering the definition of the control law from Eq. (10) and by having in mind that $\frac{\partial y}{\partial x}$ is automatically provided by ASWING code [16], the user only needs to provide $f_c(x_c, u, y)$, $h_c(x_c, u, y)$, $\left[\frac{\partial f_c}{\partial X_c}\right]$, $\left[\frac{\partial f_c}{\partial y}\right]$, $\left[\frac{\partial h_c}{\partial X_c}\right]$ and $\left[\frac{\partial h_c}{\partial y}\right]$ With thoses intels, one can recover both residual forms of the controller defined in Eq. (10)

$$\begin{aligned} r_D &= \delta - h_c(x_c, u, y) \\ r_{X_c} &= \dot{x_c} - f_c(x_c, u, y) \end{aligned} \tag{18}$$

and its associated Jacobian matrices:

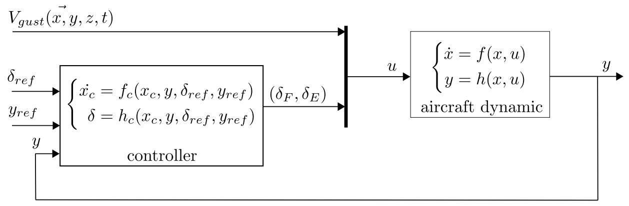
$$\left[\left[\frac{\partial r_{X_c}}{\partial Q,\dot{Q}}\right]_i \quad \left[\frac{\partial r_{X_c}}{\partial V,\dot{V}}\right]_i \quad \left[\frac{\partial r_{X_c}}{\partial P,\dot{P}}\right]_i \quad \left[\frac{\partial r_{X_c}}{\partial D,\dot{D}}\right]_i\right] = -\frac{\partial y}{\partial x}\frac{\partial f_c}{\partial y}$$

$$\left[\frac{\partial r_{X_c}}{\partial X_c,\dot{X_c}}\right]_i = \left[k_0 I - \frac{\partial f_c}{\partial X_c}\right]$$

$$\left[\left[\frac{\partial r_D}{\partial Q,\dot{Q}}\right]_i \quad \left[\frac{\partial r_D}{\partial V,\dot{V}}\right]_i \quad \left[\frac{\partial r_D}{\partial P,\dot{P}}\right]_i \quad \left[\frac{\partial r_D}{\partial D,\dot{D}}\right]_i\right] = -\left[\frac{\partial y}{\partial x}\frac{\partial h_c}{\partial y}\right] + \begin{bmatrix}0 & 0 & 0 & I_{n_D}\end{bmatrix}$$

$$\left[\frac{\partial r_D}{\partial X_c,\dot{X_c}}\right]_i = \left[\frac{\partial h_c}{\partial X_c}\right]$$

$$\left[\left[\frac{\partial r_Q}{\partial X_c,\dot{X_c}}\right]_i \left[\frac{\partial r_V}{\partial X_c,\dot{X_c}}\right]_i \left[\frac{\partial r_P}{\partial X_c,\dot{X_c}}\right]_i\right]^T = \begin{bmatrix}0 & 0 & 0\end{bmatrix}^T \tag{19}$$



$V_{gust}(\vec{x}, y, z, t)$
δ_{ref}
y_{ref}
y
$\begin{cases}\dot{x_c} = f_c(x_c, y, \delta_{ref}, y_{ref}) \\ \delta = h_c(x_c, y, \delta_{ref}, y_{ref})\end{cases}$
controller
(δ_F, δ_E)
u
$\begin{cases}\dot{x} = f(x,u) \\ y = h(x,u)\end{cases}$
aircraft dynamic
y

Figure 3: Proposed closed loop in Aswing including advanced control laws

3.3 *Linear Pattern*

Equations (19) and (10) are the milestone of the ASWING control laws extension but they still need to be implemented in the source code. Therefore one proposes a linear pattern depicted in figure 4. The user only needs to provide every A,B,C,D matrices of each block in a MATLAB format. If a block is not used, it will be automatically removed from the program. Regarding the pattern itself, one can implements sensors and actuators dynamics and saturations, a full MIMO linear controller, and an anti-windup system. Moreover, every block can be bi-scheduled by every measurable output. Furthermore, noise can be added to the simulation for temporal robustness studies in a stochastic environment. This pattern is motivated by the fact that UAVs are using small embedded sensors and actuators leading to a noisy environment, saturations delays, etc. The authors consequently thought it would be relevant to consider them directly in a numerical tool. Let $x_c = (x_A, x_S, x_{AW}, x_K)^T$ be the control states vector, following Eq.(18), In its full configuraiton (every block used) one has :

$$h(x_c, y, u) = sat(C_A x_A) \tag{20}$$

where $sat(C_A x_A)$ $sat(C_S x_S)$ are the saturation function associated to actuators and sensors
Moreover, one has

$$\frac{\partial h(x_c, y, u)}{\partial x_c} = \begin{pmatrix} \frac{\partial sat(C_A x_A)}{\partial x_A} & 0 & 0 & 0 \end{pmatrix} \tag{21}$$

The jacobian matrix becomes :

$$\frac{\partial h(x_c, u, y)}{\partial y} = 0_{n_\delta, n_y} \tag{22}$$

At the light of figure 4 f_c expression comes:

$$f(x_c, y, u) =$$

$$\begin{pmatrix} A_a x_A + B_A C_K x_K + B_A D_{K,Q} sat(C_S x_S) \\ A_S x_S + B_S(y + w_y) \\ A_{AW} x_{AW} + B_{AW}(C_A x_A - sat(C_A x_A)) \\ A_{KxK} + B_{K,Q} sat(C_S x_S) - B_{K,Q} Q^* \\ +B_A \delta^* \\ \dots \\ \dots \\ B_{K,AW}(C_{AW} x_{AW} + D_{AW}(C_A x_A - sat(C_A x_A)) \end{pmatrix} \tag{23}$$

With its associated jacobians:

$$\frac{\partial f}{\partial x_c} = \begin{pmatrix} A_a & B_A D_{K,Q} \frac{\partial sat(C_S x_S)}{\partial x_S} & O_{n_A,n_{AW}} & B_A C_K \\ 0_{n_s,n_A} & A_S & 0_{n_S,n_{AW}} & 0_{n_S,n_K} \\ B_{AW} C_A - B_{AW} \frac{\partial sat(C_A x_A)}{\partial x_A} & 0_{n_{AW},n_S} & A_{AW} & O_{n_{AW},n_K} \\ D_{AW}\left(C_A - \frac{\partial sat(C_A x_A)}{\partial x_A}\right) & B_{K,Q} \frac{\partial sat(C_S x_S)}{\partial x_S} & B_{K,AW} C_{AW} & A_K \end{pmatrix} \tag{24}$$

And

$$\frac{\partial f}{\partial y} = \begin{pmatrix} 0_{n_A,n_y} \\ B_S \\ O_{n_{AW},n_y} \\ O_{n_K,n_y} \end{pmatrix} \tag{25}$$

Adding 23, 24, 25, 20, 21 and 22 to 19 recovers 10 for the linear pattern case.

Interpolation methods:

Interpolation methods have been implemented to set up the A,B,C and D matrices of each block regarding their discrete scheduled values. The first method is the nearest neighbor, the second a bilinear interpolation and the third is a 2D polynomial interpolation. Interpolation methods are often used as gain scheduling methods, however, a stability study must be made to ensure that the flight envelope is not too coarse to threaten the stability of the overall closed-loop system during working point switches. ASWING modification does not provide such an analytical tool. The user will consequently make sure that the aircraft flight envelope has enough points to avoid such stability problems.

4 EXAMPLES

The authors must notify, that the change in ASWING code lets the previous one unchanged meaning that the examples presented in [11] are still valid and usable. The lecturer must see this work as a conservative exension.

Span	4.4m	Time range	36h
Weight	14kg	Range	3000km
Cruise speed	24m/s	nominal altitude	100m

Table 2: Mermoz main specs

4.1 *open loop control behavior with actuator dynamic:*

The first example aims at showing the effect of an added dynamic and saturations to an aircraft actuator such as an engine. The aircraft used in the next examples is the Mermoz UAV hydrogen prototype [4] shown on figure 5 whose main characteristic are depicted on table 2. The engine limits are arbitraly set to 0 and 7N for a non negative thrust behavior and limited engine. A first order linear system with a time response of 3s is used to recover its dynamic. To ensure that the simulation would not crashed, the aircraft is cantilevered to the ground. The figure 6 shows the benefit of added pattern to take into account actuators dynamic and saturations. The black line accounts for the previous way, and the blue the new one.

4.2 *Output feedback Longitudinal controller for gust harvesting: effect of actuators or sensors dynamics*

As Mermoz is supposed to fly around 100m over the atlantic ocean, one plans to harvest or alleviate the turbulence created by the sea waves. Futhermore as the altitude is quite low, the aircraft trajectory must be holden in a given range rising a control law problem. [6] [5] have shown that there exists control law stategies for small UAVs which lead to power saving using the surrounding aircraft environement. They firstly shown that for a single pulsation sinewave vertical gust, one could harvest energy and witness a positive gain in altitude. The proposed control law was a PD control using the encountered vertical gust velocity measurement as input. In this example, one proposes a simple propotionnal feedback of the aircraft pitch angle as a horizontal stab control law, leading to a gain in altitude. However one presents the effect of actuator or sensor dynamic on the performance. 3 different first order dynamics have been studied with a time response of : 0.1 s 1s and 10s (respectively blue, orange and yellow on figure 7). The vertical gust is a 1Hz sinewave with hyperbolic tangent activation function, the amplitude is 1.5m/s. The feedback gain is set to -1.8. The figure 7 shows that for slow actuators one witness a delayed flap answer with smaller amplitude. However the performance in gain of altitude seems to be better with a slow actuator. This means that the optimum feedback gain used for the simulation should be smaller. Consequenlty if the actuator dynamic response is slower or near the aircraft one, it is highly possible that the control law will witness a change in its performance.

4.3 *Performance robustness to a full bandwith gust model:*

The last example follows the same protocole as the previous example. However the authors has extended the gust

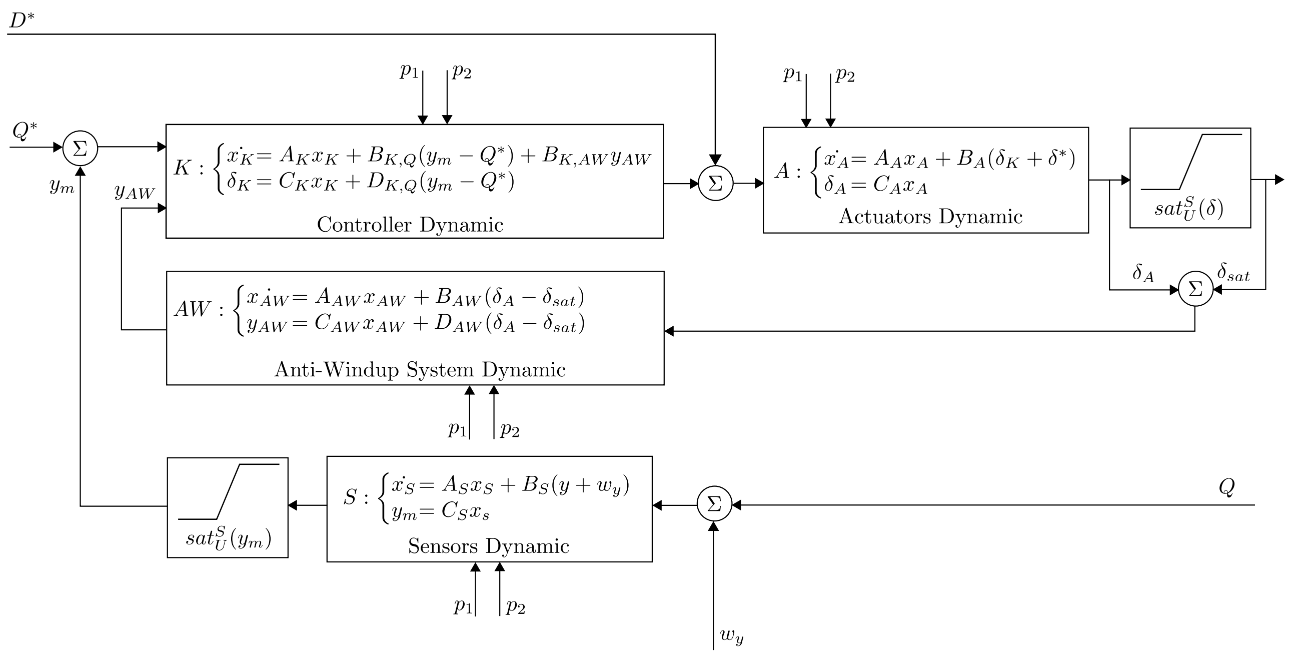

D^*
Q^*
Σ
y_m
y_{AW}
p_1 p_2
$K : \begin{cases} \dot{x_K} = A_K x_K + B_{K,Q}(y_m - Q^*) + B_{K,AW} y_{AW} \\ \delta_K = C_K x_K + D_{K,Q}(y_m - Q^*) \end{cases}$
Controller Dynamic
$A : \begin{cases} \dot{x_A} = A_A x_A + B_A(\delta_K + \delta^*) \\ \delta_A = C_A x_A \end{cases}$
Actuators Dynamic
$sat_U^S(\delta)$
δ_A
δ_{sat}
$AW : \begin{cases} \dot{x_{AW}} = A_{AW} x_{AW} + B_{AW}(\delta_A - \delta_{sat}) \\ y_{AW} = C_{AW} x_{AW} + D_{AW}(\delta_A - \delta_{sat}) \end{cases}$
Anti-Windup System Dynamic
$sat_U^S(y_m)$
$S : \begin{cases} \dot{x_S} = A_S x_S + B_S(y + w_y) \\ y_m = C_S x_s \end{cases}$
Sensors Dynamic
Q
w_y

Figure 4: Linear control pattern proposed

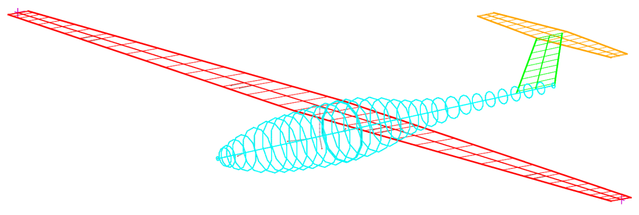


Figure 5: Mermoz aircraft: ASWING configuration

model function of ASWING with a full 3D spatio temporal Von Karman turbulence model based on the adaptation to UAV work of []. The figure 8 clearly show that the control does not satisfy the energy harvesting objectives no matter the dynamic of the actuator. One must consequently seek for a better and more robust control law to fullfill the objective for an extended gust bandwith.

5 CONCLUSION

This article has presented a methodology to modify ASWING source code so that control laws with internal dynamics commonly used by the control theory community can be implemented. One has presented the different algorithms used in ASWING and the proper modifications that have to be made to fulfill this objective. Some theoretical results have been recalled for a better understanding of the methodology. Moreover, one has to make sure that the modifications brought do not lead to an added stiffness on the differential equations leading to the solver divergence. Finally, the modification has been illustrated with 2 examples to have a quick partial view of the modification. From this modification, the users can use this methodology to quickly implement and benchmark sophisticated bi-scheduled controls law such for example Linear quadratic Gaussian, H_∞ / H_2, Guardian mapping, Fractional order controller, $\mu analysis$. He can also investigate sensors and actuators' saturations effects and implement anti-wind-up and or observers. Sensors noises can also be taken into account for robustness study to stochastic environment. Future works aims at providing a "craftable fyable aircraft" design tool which respects ASWING theoretical background for automatic and parametric design of UAVs based on modern crafting technics

REFERENCES

[1] Julian Theis, Harald Pfifer, and Peter Seiler. Robust Modal Damping Control for Active Flutter Suppression. *Journal of Guidance, Control, and Dynamics*, 43, March 2020.

[2] David K. Schmidt, Brian P. Danowsky, Aditya Kotikalpudi, Julian Theis, Christopher D. Regan, Peter J. Seiler, and Rakesh K. Kapania. Modeling, Design, and Flight Testing of Three Flutter Controllers for a Flying-Wing Drone. *Journal of Aircraft*, 57(4), July 2020.

[3] David Schmidt, Brian Danowsky, Peter Seiler, and Rakesh Kapania. *Flight-Dynamics and Flutter Analysis and Control of an MDAO-Designed Flying-Wing Research Drone*. January 2019.

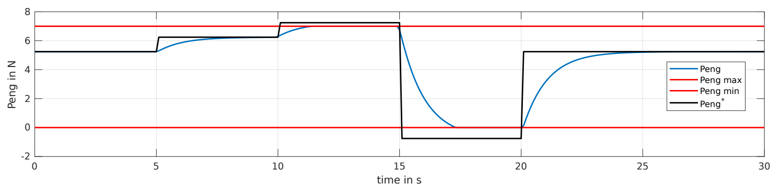


Figure 6: Effect of first order dynamic on actuators

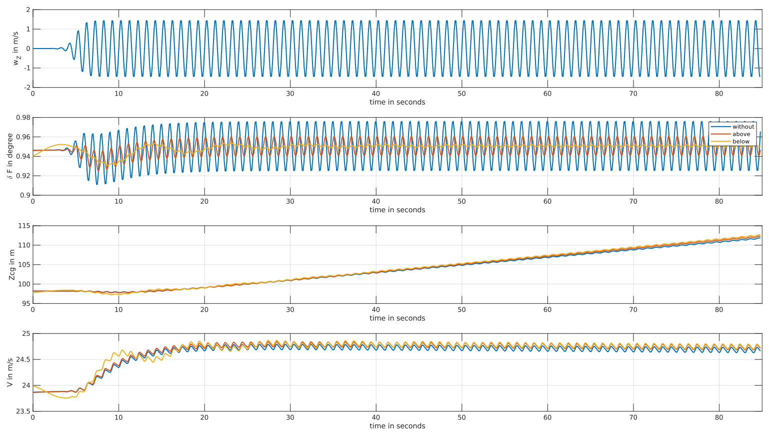


Figure 7: Single pulsation gust soaring and effect of the actuator dynamic on performance

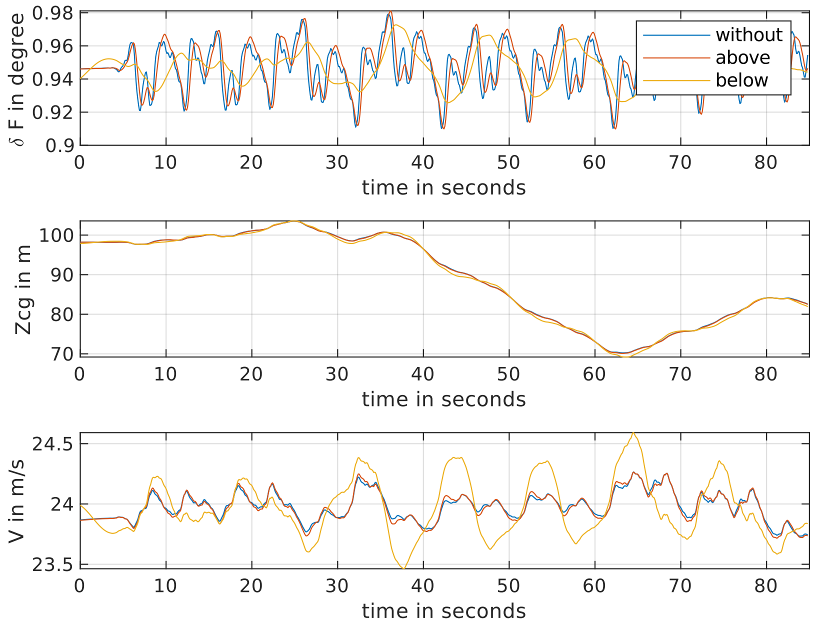


Figure 8: Robustness of gust soaring strategy for an extented bandwidth

[4] N Gavrilovic, D Vincekovic, and J-M Moschetta. A Long Range Fuel Cell/Soaring UAV System for Crossing the Atlantic Ocean.

[5] Nikola Gavrilovic, Emmanuel Benard, Philippe Pastor, and Jean-Marc Moschetta. Performance Improvement of Small Unmanned Aerial Vehicles Through Gust Energy Harvesting. *Journal of Aircraft*, 55(2), March 2018.

[6] N Gavrilovic, A Mohamed, M Marino, S Watkins, J-M Moschetta, and E Benard. Avian-inspired energy-harvesting from atmospheric phenomena for small UAVs. *Bioinspiration & Biomimetics*, 14(1):016006, November 2018.

[7] Ravi Jaiswal, Abhishek Shastry, Swati Swarnkar, and Mangal Kothari. Adaptive Longitudinal Control of UAVs with Direct Lift Control. *IFAC-PapersOnLine*, 49(1), January 2016.

[8] R. F. Latif, M. K. A. Khan, A. Javed, S. I. A. Shah, and S. T. I. Rizvi. A semi-analytical approach for flutter analysis of a high-aspect-ratio wing. *The Aeronautical Journal*, 125(1284), February 2021. Publisher: Cambridge University Press.

[9] Wang Rui, Zhu Xiaoping, and Zhou Zhou. Design Gust Alleviation Controller for Highly Flexible Solar UAV. In *2011 Third International Conference on Measuring Technology and Mechatronics Automation*, volume 1, pages 930–933, January 2011. ISSN: 2157-1481.

[10] Ya Wang and Daniel J. Inman. Autonomous Gust Alleviation in UAVs. In *Advanced UAV Aerodynamics, Flight Stability and Control*. John Wiley & Sons, Ltd, 2017.

[11] Mark Drela. Integrated simulation model for preliminary aerodynamic, structural, and control-law design of aircraft. In *40th Structures, Structural Dynamics, and Materials Conference and Exhibit*, St. Louis,MO,U.S.A., April 1999. American Institute of Aeronautics and Astronautics.

[12] Mateus d Virgilio Pereira, Ilya Kolmanovsky, Carlos E. Cesnik, and Fabio Vetrano. Model Predictive Control Architectures for Maneuver Load Alleviation in Very Flexible Aircraft. In *AIAA Scitech 2019 Forum*. American Institute of Aeronautics and Astronautics. _eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2019-1591.

[13] Stuart P. Andrews. Modelling and simulation of flexible aircraft : handling qualities with active load control. March 2011. Accepted: 2012-12-13T11:36:23Z Publisher: Cranfield University.

[14] Xuerui Wang, Tigran Mkhoyan, and Roeland De Breuker. Nonlinear Incremental Control for Flexible Aircraft Trajectory Tracking and Load Alleviation. *Journal of Guidance, Control, and Dynamics*, May 2021. arXiv: 2101.00594.

[15] Mark Drela. ASWING 5.86 Technical Description — Steady Formulation.

[16] Mark Drela. ASWING 5.81 Technical Description — Unsteady Extension.

[17] Mark Drela. *2D transonic aerodynamic design and analysis using euler equations*. PhD thesis, MIT, 1983.

[18] Eugene Isaacson and Herbert B. Keller. *Analysis of numerical methods*. Dover, New York, NY, 1994. OCLC: 30032279.

Acknowledgements

This work has not been fulfill in partnership with the main developter of ASWING namely Prof M.Drela from MIT, as it is not a open source and free to use software, one can not let the source code accessible. One strongly aims at proposing this modification as an official ASWING toolbox.

Field report: deployment of a fleet of drones for cloud exploration

Gautier Hattenberger*,[1], Titouan Verdu[1], Nicolas Maury[2], Pierre Narvor[4], Fleur Couvreux[2], Murat Bronz[1], Simon Lacroix[4], Grégoire Cayez[5], and Gregory C. Roberts[2,3]

[1]École Nationale de l'Aviation Civile, Université de Toulouse, Toulouse, France
[2]Centre National de Recherches Météorologiques, Université de Toulouse, Météo-France, CNRS, Toulouse, France
[3]Scripps Institution of Oceanography, University of California San Diego, La Jolla, USA
[4]Laboratoire d'Analyse et d'Architecture des Systèmes, Université de Toulouse, CNRS, Toulouse, France
[5]École Nationale de la Météorologie, Météo-France, Toulouse, France

Abstract

Drones are commonly used for many civil applications and the procedures to operate them have evolved during the past years to make them accessible to those with limited piloting skills in several scenarios. However, the deployment of a fleet in the context of scientific research can lead to complex situations that require an important preparation in terms of logistics, permission to fly from authorities, and coordination during the flights. This paper is a field report of the flight campaign held end of January 2020 at the Barbados Island as part of the NEPHELAE project. The main objectives of the project were to fly into trade wind cumulus clouds to understand the microphysical processes involved in their evolution, as well as to provide a proof of concept of sensor-based adaptive navigation patterns to optimize the data collection. After presenting the overall flight strategy and the context of operation, the main challenges and the solutions to address them will be presented, to conclude with the evaluation of some technical evolution developed from these experiments.

1 Introduction

Drones are commonly used for many civil applications and the procedures to operate them have evolved during the past decade to make them accessible to those with limited piloting skills in myriad scenarios. However, the deployment of a fleet in the context of scientific research can lead to complex situations that require an important preparation in terms of logistics, permit to fly from authorities and coordination during the flights. This paper is a field report of the flight campaign held end of January 2020 at the Barbados Island as part of the NEPHELAE project.

The context and main goal of the project are detailed in section 2. After presenting some of the preparation and logistic constraints, the overall flight strategy, its challenges and the solutions to address them will be presented. To conclude a summary of the flights will be provided with the evaluation of some technical evolution developed from these experiments.

*Corresponding author: gautier.hattenberger@enac.fr

2 Context

The purpose of the NEPHELAE project is to develop and deploy a fleet of autonomous drones to collect data within and around cumulus clouds. The final objective for the atmospheric scientists at the Centre National de Recherches Métérologiques (CNRM) that are leading the project is to better understand the mixing processes during the evolution of these clouds. The main region of interest for these observations is the border of the cloud that needs to be sampled at the frontier between the open air and the water-saturated region. Several publications have already described the flight patterns specifically designed for this task [1, 2] and showed the advantage of these strategies in cloud exploration in a simulated cumulus field [3, 4].

As a summary, the general idea is to use the real-time data from the on-board sensors to decide whether or not the plane is inside the cloud or not. Based on this decision and the general flight strategy (border exploration, border+center exploration, . . .) a sequence of arcs and straight lines is performed. Figure 1 presents the area of interest and some of the possible flight patterns.

The key element of this project is the operation of several drones at the same time with several objectives: increase the spatial and temporal sampling resolution by deploying a network of sensors, extend the observational footprint of the in-situ observations using advanced mapping techniques, and perform synchronized measurements of the same volume at different locations to estimate the transport of water and heat inside and surrounding the cloud.

Finding the right location to carry out the flights is already a challenge. The target clouds are trade wind cumuli, which means that the flights will be over the sea and in the tropics. The prevailing winds average 8 m/s and it is required to follow the same cloud as long as possible (the life-time of a cloud is around 20 to 30 minutes), it means that the flight

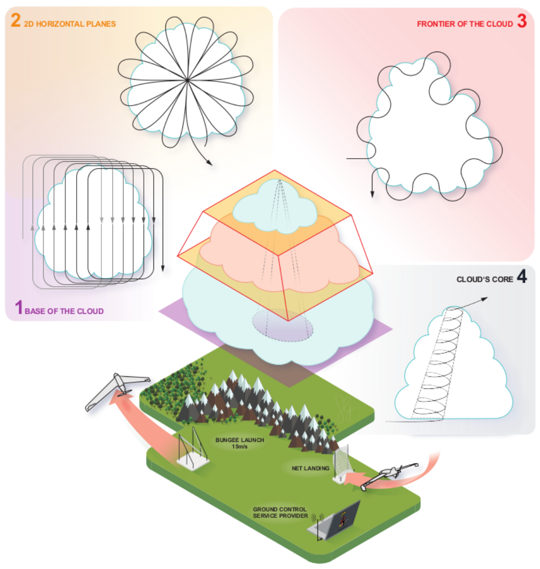


Figure 1: Cloud interest zones (illustration and design by Sarah Gluschitz)

area has to be large enough, and the flight ceiling at least 1500 to 2000 meters above sea level (ASL). Several places have been considered, and visited by a team member prior to the mission, depending of their weather, accessibility, local regulation, etc.

We finally had the opportunity to participate to an international flight campaign, called EUREC4A [5, 6], at the beginning of 2020 on the island of Barbados. This campaign already involved boats, piloted planes and two other types of drones. It is important to note that UAV flights are strictly forbidden by Barbados authorities, except where specifically allowed in the context of this scientific mission, as long as the drones have permit to fly from authorities from their own country. Since the selected location for drones was the same for all teams, the airspace had to be shared.

The next sections details the flight operation and the required preparation and logistics for a long mission abroad.

3 Preparation and logistics

The selection of the flight location was complex as several factors had to be considered. The first constraint was to be located on the east coast as the prevailing wind is blowing west from the ocean and the goal is to catch the clouds before they reach the island. The international airport is located at the south end of the island, which means that our flight area should be north to avoid conflicts with the planes during their initial climb. Finally, for fixed-wing operations the field should be flat, far enough from populated area and void of surrounding obstacles. None of the three possible spots perfectly matched all the criteria, but the one called Morgan Lewis (see figure 2) was the best option with a long field well-oriented toward the sea and with few obstacles, except for hills on the side and a downward slope that made the automatic landing impossible. The landings were finally all performed by the safety pilot.

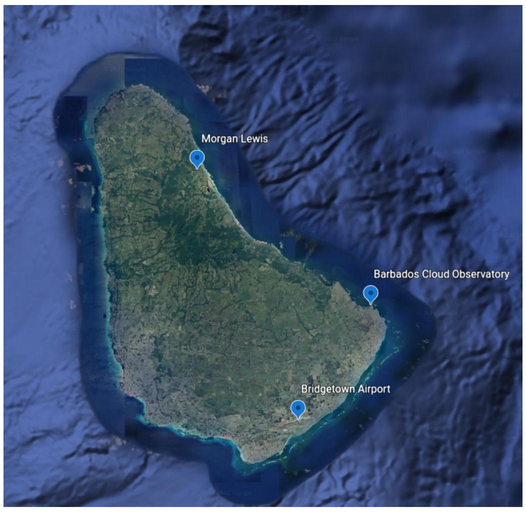


Figure 2: Map of Barbados Island with flight location at Morgan Lewis, Barbados Cloud Observatory (BCO) and Bridgetown Airport (BGI)

A total of 10 people (researchers, engineers, PhD students) took part of the mission for a duration between 1 week to almost 4 weeks. The planes, ground equipment and computers, scientific sensors and batteries were sent two months in advance by ship. Containers served as the base station for operation and storage as seen in the figure 3.

Figure 3: View of the temporary operation center and storage

4 Flight operation strategy

The flight operation strategy has been defined during the preparation time before the mission to take into account the needs and the maturity of the technology being deployed. The details of the overall software architecture and the algorithms can be found in [1]. As a summary, during a typical flight five operators were working together each with a specific role:

- The *atmospheric scientist* is monitoring the real time sensor values collected by the UAVs. When he estimates that a UAV is crossing a cloud worth sampling, he requests to deploy a specific pattern to the mapmaker operator. He is also in charge of the regional weather forecast and near real-time satellite images to determine if clouds are coming within the next hour.

- The *mapmaker operator* is checking the real-time mapping process based on Gaussian Process Regression. Once he receives new instruction from the atmospheric scientist, he creates a new mission element with the desired parameters.

- The *UAV operator* is controlling the flights from the Ground Control Station (GCS). In particular, he is in charge of take-off, landing and waiting procedures, as well as the general safety of the flights.

- The *flight director* is the coordinator of the three other operators. He is checking the created mission and will decide if they should be accepted or rejected. He is also in charge of the coordination with the other teams sharing the airspace and is the point of contact for the local Air Traffic Control.

- The *safety pilot* is outside with the remote control and is handling the planes. He can take back the control when flying in line-of-sight and is piloting for the very last part of the landing.

Figure 4 shows the operation center with a display for the atmospheric scientist to confirm cloud detection (left), the operator in charge of the real-time mapping system (center), and the UAV operator (right). The safety pilot remained outside during take-off and landing operation, and the flight director stands behind the other operators to have a global view of the flight operations.

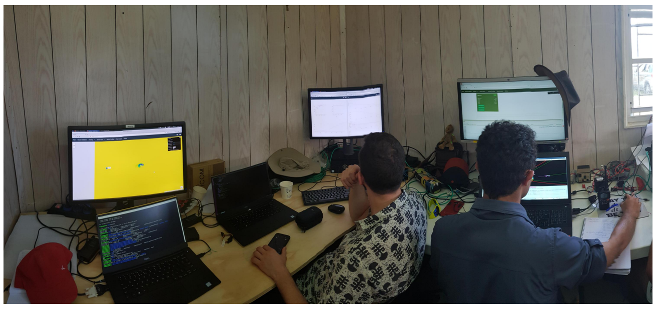


Figure 4: Operation center with from the left: atmospheric data and mapping display, map operator and UAV operator

The role of the safety pilot is always to guarantee the safety of the operations during critical phases (take-off and landing, flight near populated area, . . .). When dealing with several aircraft, the problem is even more complex. The pre-campaign configuration of the Paparazzi UAV is that each drone has a dedicated pilot and remote control receiver (RC). However, considering the initial goal of the project to have up to 4 or 5 UAVs at the same time, it was not a viable option as the number of qualified safety pilots would not have been sufficient. In addition, previous experiences have shown that the risk of mixing the RC transmitters is real and has lead to catastrophic situations. The solution that has been selected for this project was to use a single safety pilot with only one controller. All planes were bonded to the same RC and a special software tool was developed for the UAV operator to select which plane is being controlled at a particular time. This does not go without risks – if a plane is selected in the wrong mode, it might enter to a safety mode and go back home. To reduce this risk, the RC selector was also checking the status of the autopilot (flight and RC mode) to decide if the selection of a particular UAV is valid or not.

5 Deployment and practical constraints

5.1 *Airframe and ground equipment*

The requirements for the plane to be used for the NEPHELAE project were particularly hard to meet. A prototype of custom aircraft had been designed [7], conforming to the requirements in terms of flight performances (flight speed and time, altitude and range, payload capacity). However, the result of the high-performance design was a UAV that was too fragile and complex to operate in a scientific field experiment, without proper facilities for maintenance and repairs. The decision was finally made to used commercial foam airframe, called X6, modified to integrate the scientific payload as seen in figure 5. They have been used in previous missions [8, 9, 10] and proven to be robust and easy to repair.

The main impact was a limited flight time of 1 hour (plus 10 minutes for margins) instead of more than 2 hours for the custom design.

The post-campaign analysis and further flight testing have shown that it was a good decision to give the priority to the robustness of the aircraft as the repeated landings and occasional mishaps would have put the 3D-printed high-performance planes beyond repair. The flight time was indeed a limitation but was still compatible with the allowed airspace dimensions. A better integration of the batteries, sensors , flight controller, optimized flight performance (cruise airspeed, climb rates, etc,) would have allowed to increase this flight time to 1.5 hours. In addition, a cloud-targeting system would also reduce the time required to intercept a cloud – allowing more time for the scientific part of the mission.

In total, 4 planes were available, but only 3 flew, with at most 2 planes at the same time. This was much less than the initial plan, involving up to 4 or 5 planes. Several reasons can be found. One was that the time to operate the drones on the ground, with a bungee launch is quite long, as well as the time to reach the altitude and position to start tracking the clouds.

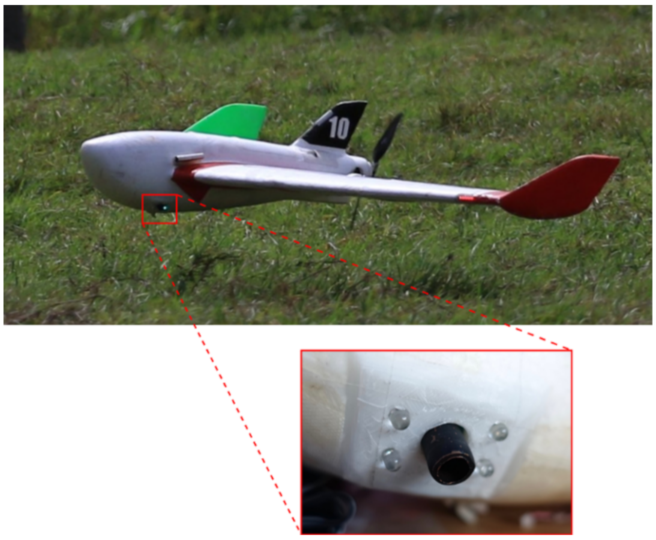


Figure 5: Skywalker X6 with integrated sensor for cloud detection and measurements, details in [8, 9]

Using more than 2 planes would have resulted in a very short time of effective simultaneous cloud exploration. One of the main issues was that the automatic, or at least assisted, task allocation algorithm had not been sufficiently developed prior to the experiment in Barbados. Priority had been given to the real-time cartography and adaptive flight patterns. As a result, the burden for conflict avoidance and synchronization was left on the shoulder of the UAV operator, in addition to the usual flight parameter monitoring. Conducting the ensemble of the missions requires training and experienced UAV operators in conjunction with a safety pilot, especially for the take-off and landing phases of the mission.

Concerning communications, the 2.4GHz long range modems *P2400* from *Microhard* have been tested and proven themselves to be reliable, with a constant data flow up to 14km from the base station (equipped with a directed antenna and set to the maximum power).

5.2 *Flight plan*

The flight space has been organized as shown in Figure 6. The red trapezoidal zone is the allowed flight area, defined with the Barbadian Civil Aviation Authorities. This flight zone is trapezoidal to account for variability in the prevailing wind, and also inhibits the UAV from exiting the flight zone in case of a *Return to HOME* procedure. The flight ceiling was limited to 1000 m ASL because our airspace was below the approach zone for the international airport (BGI). On several occasions, a 2000 m ASL ceiling was requested on a per-use basis and only in the afternoon. The maximum allowed distance from the GCS (pink arc) is 15 km.

The Green area (figure 6) is the normal operation area inside which adaptive navigation patterns were used based on the ground operator's instructions. If a plane deviated from this area while tracking a cloud, it was automatically assigned to a standby safe position close to the sea shore. Finally, the orange rectangle represents the flight airspace used by a team from the University of Colorado to fly a small drone below the cloud base [11].

Since we are flying inside clouds, therefore above cloud base, it is possible to fly together, but with great care. In general, this space was avoided when the two teams were operating simultaneously . Another large drone (4m wing span, 25 kg; BOREAL SAS, Toulouse, France) was operated from the same field for long distance missions. All operations with light drones where forbidden during takeoff and landing of this drone.

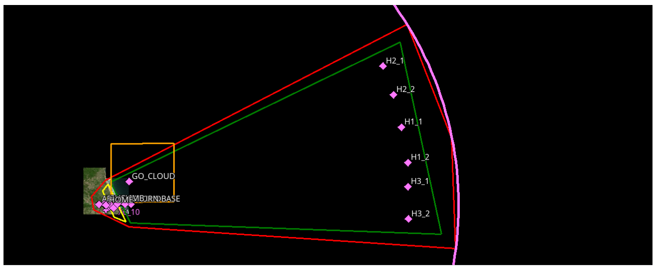


Figure 6: Flight zones: limit of the allowed area (red), normal operation area (green), shared airspace with another team (orange)

Figure 7 presents the steps in a typical flight:

Step E1 Take off, initial climb and waiting on a circle.

Step E2 Go to a climb point above the sea and reach the final altitude, defined by the altitude of cloud base and the objectives of the flight. Cloud base had been determined prior to the mission using real-time observations from remote sensing instruments at the Barbados Cloud Observatory (BCO). Flight altitudes were at least 100 m above cloud base.

Step E3 Reaching the eastern end of the flight zone at final altitude. During this time, the cloud sensor offset is computed from background noise outside of clouds. When reaching the search zone, the planes are performing hippodromes (10 to 14 km from the GCS) perpendicular to the prevailing wind until a cloud is detected or mission is aborted.

Step E4 When a cloud is detected, the adaptive flight patterns are activated to track the cloud and achieve the required measurements.

Step EF The final step is the descent to waiting circle and landing under the supervision of the safety pilot.

In this example (Figure 7), two planes have been deployed at the same time, at two different altitudes (50 meters of vertical separation). Only the yellow plane was able to catch and track a cloud. The second plane was too far to join the first one and went back home directly. While only one UAV tracked the cloud during this mission, there were two missions during which both UAVs tracked the same cloud simultaneously.

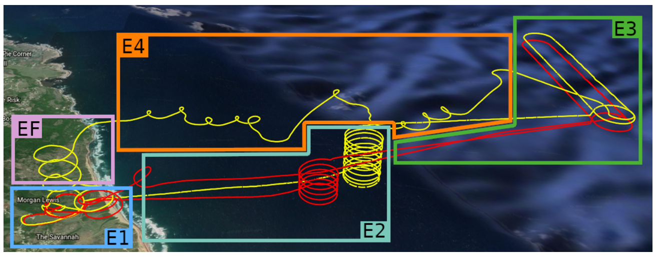


Figure 7: The different steps a typical mission

5.3 *Improvements and corrections*

Despite several preparation flights before the actual campaign, it was not possible to anticipate all the possible issues. Ideally, the entire system including the sensors, adaptive sampling strategies and mapping software would have been tested in marine clouds near the coast of France prior to deployment in Barbados; however, delays in the fabrication and instrumentation of the UAVs, as well as necessary changes in the adaptive sampling strategy and difficulties in securing an authorized airspace rendered a complete demonstration of the system prior to the experiment impossible. Consequently, several modifications to the setup in the field were needed to improve the safety and efficiency of the flight.

One of the unforeseen issues to solve concerned the real-time interpretation of the data from the cloud sensor. While the cloud sensor had been used in prior field experiments [8], it had not been possible to test the sensor in conjunction with the adaptive sampling strategy prior to the experiment in Barbados. For example, the signal is sensitive to the input voltage, which also varied with the throttle commanded by the autopilot – the inline voltage regulators were insufficient for this application of the cloud sensor and hardware modifications were not technically possible. During the turns, the autopilot increased the throttle to maintain the prescribed altitude and created artificial peaks in the cloud sensor. It was anticipated that both a hysteresis filter and low-pass filter would be needed to correctly detect the cloud edge. However, the correlation with motor power consumption made the sensor more challenging to use during the adaptive sampling phase of the mission. Correction factors proportional to the battery voltage have then been added during the mission. Several calibration flights were needed to achieve a proper calibration, delaying the schedule for scientific operations.

Another issue not previously anticipated was the location of the cloud sensor, which initially pointed horizontally out the side of the fuselage. During previous experiments, anomalies related to direct sunlight on the cloud sensor could easily be removed in post-processing. However, during the flight campaign in Barbados, direct sunlight on the cloud sensor also interfered with the adaptive sampling. Fortunately, the foam structure allowed to move the sensor to look 45 degrees down as seen in Figure 5 and no further issues were reported.

In addition, several flights have been interrupted or even resulted in a mishap due to bad GPS reception. To mitigate this issue, the GPS receiver was moved outside of the fuselage on an metallic sheet connected to the battery ground wire. Tape was used to protect the electronic from humidity and water droplets while sampling inside the clouds.

Finally, the transition from the open ocean to the hilly terrain generated turbulent structures that caused the UAVs to stall during landing procedures. After around a dozen flights, we settled on a landing corridor that seems to generate less turbulence and increased the airspeed to penetrate the areas that were turbulent. The safety pilots also initiated the landing procedures prior to turning for the final approach.

6 Summary and improvements for future operations

6.1 *Flights summary*

Table 1: Summary table of the flights during the experimental Barbados flight campaign

Number of flights	48	22 flights realized with two UAVs
Data recorded	45 hours	Flight time average around 53 minutes per flight
Calibration flights	23	Cloud sensor and UAV calibration, validation of the flight pattern
Measurement flights	25	Vertical profile and cloud tracking mission
Viable scientific flights	18	Autonomous tracking of a cloud during more than 2 minutes. Average following time around 5 minutes per tracking.

Table 1 presents a summary of the flights during the Barbados campaign. Each ‘flight‘ refers to a complete operation involving one or more drones at the same time. The only attempt to fly with three drones was aborted due to the unsafe behavior of one of the planes.

The first remark is that the number of calibration/validation flights comprised nearly half of the total number of flights. As stated previously, complete testing of the system including the sensors, adaptive sampling strategies and mapping software in real clouds could not be done before the experiment, and a number of issues had to be addressed once operations began in the field.

The second point is that only 18 flights were able to track a cloud for more than 5 minutes. As there were no dedicated cloud targeting systems implemented during this campaign, the UAVs orbited in a hippodrome pattern perpendicular to the prevailing wind up to 10 km from the GCS until intercepting a cloud by chance and start the tracking. Fortunately, the altitude of cloud base was well identified using remote sensing information from BCO. Nonetheless, a special device, called *AllSky*, with two cameras a few hundreds of meters apart could have provided the initial position of the clouds [12] allowing multiple UAVs to immediately intercept the cloud and start the scientific mission.

Some flights have also aborted after loosing the track of the cloud as the environment is very dynamic at the border of the cloud. When trying to turn back inside the cloud, if the shape is too different from the previous turn, the plane may continue on its circle without reentering the cloud again.

Out of the 18 usable flights, 6 were used to process meaningful data to build diagrams about the exchange of heat and water between the cloud and the surrounding atmosphere. This includes a flight with 2 drones tracking the same cloud with the adaptive patterns designed for the project. Given that cloud cover is usually less than 10 percent, the fact that we could track any clouds at all is quite an accomplishment.

6.2 Lessons learned and possible improvements

The NEPHELAE flight campaign was ultimately a success as we managed to deploy several drones, with novel adaptive flight patterns to autonomously track clouds for the first time and gather scientific data, which are being exploited to improve the understanding of entrainment mixing and cloud evolution. However, several points could be improved and lessons learned form the difficulties faced during the project.

The first one is that the robustness of the plane and the possibility to repair in the field is extremely important, even more than adhering to flight performance objectives. In the case of optimizing the airframe design for NEPHELAE, accounting for desirable conditions in term of flight time, airspeed and payload capacity, led to an interesting prototype, but unable to operate in field campaign conditions. An other aspect of the robustness is the quality of the electronic boards integration, both for payload and autopilot, which is particularly important when conducting a field experiment where the salt from the sea spray corrodes electrical components. The malfunctions with the GPS receivers and some issues with the cloud sensor were related to insufficient voltage regulation, corrosion or damaged cabling.

The second issue to mention is the robustness of the autonomous navigation patterns, which should be able to recover and continue the adaptive sampling after losing track of a cloud. This issue has been addressed after the campaign and validated with hybrid flights (real planes flying in simulated clouds) at the Centre de Recherches Atmosphériques (Lannemezan, France). After running the planes in different scenarios, they were able to relocate the border and continue the exploration.

In one occasion, the use of the RC switching mechanism to fly multiple aircraft from the same transmitter almost led to a mishap, because the UAV operator selected a plane already flying in autonomous navigation while the safety pilot RC was on manual position. Fortunately, the plane was in line-of-sight of the safety pilot who managed to stabilize the flight before switching the UAV back to autonomous flight. Since the GCS tool for plane selection does not have direct feedback from the RC (except through the status of the plane), the only way to prevent a repeat of this situation is by ensuring proper dialog between the UAV operator and safety pilot to assess the currently selected mode on the RC before switching.

As discussed earlier, a dedicated cloud targeting system would have reduced the time needed to intercept the clouds and begin the scientific part of the mission. An integrated cloud targeting would guide the UAVs automatically and could have been accomplished with onboard cameras or by deploying a ground-based system such as the *AllSky* system.

Finally, the last point that needs improvement is the automatic task allocation in order to deploy a larger fleet. The workload on the different operators is already high and several tasks should be automated. This includes improvements to take-off procedures, assistance for collision avoidance and the possibility to assign high level goals to the fleet, leading to less manipulation by the UAV operator and safety pilot.

7 Conclusion

Within the NEPHELAE project, an atmospheric science driven study, a dedicated architecture has been developed to operated multiple drones during an international field campaign to follow the evolution of clouds. In addition to the usual technical challenges to fly beyond visual line-of-sight up to 14 km from the GCS and at relatively high altitude, many operational constraints had to be addressed. The overall campaign was a success considering the number of flights, and the value of scientific results that have been extracted from them. Nonetheless, several difficulties were encountered during the mission and the lessons learned will be considered for future projects. Notably, some key elements of the original architecture could not be developed in time, particularly, task planning and a cloud targeting system, which are the main focus of future work. Preliminary scientific results can be found in [4].

Acknowledgements

This research was supported by Agence Nationale de la Recherche (Project-ANR-17-CE01-0003), Aerospace Valley, Météo-France and Région Occitanie. Observation data were collected at the Pyrenean Platform for Observation of the Atmosphere P2OA (http://p2oa.aero.obsmip.fr). P2OA facilities and staff are funded and supported by the University Paul Sabatier, Toulouse, France, and CNRS (Centre National de la Rechercher Scientifique). We also acknowledge the Centre de Recherches Atmosphériques (CRA) for their assistance and allowing access to their facilities and meteorological data for the flight tests during the NEPHELAE project.

REFERENCES

[1] 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 *IROS 2020, IEEE/RSJ International Conference on Intelligent Robots and System*, 2020 IEEE/RSJ International Conference on Intelligent Robots and System, pages pp 1429–1435, Las Vegas (on line), United States, October 2020. IEEE.

[2] T. Verdu, G. Hattenberger, and S. Lacroix. Flight patterns for clouds exploration with a fleet of uavs. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 231–237, June 2019.

[3] Christophe Reymann, Alessandro Renzaglia, Fayçal Lamraoui, Murat Bronz, and Simon Lacroix. Adaptive sampling of cumulus clouds with UAVs. *Autonomous Robots*, 42(2):pp.491–512, February 2018.

[4] N. Maury, G. C. Roberts, F. Couvreux, T. Verdu, P. Narvor, N. Villefranque, S. Lacroix, and G. Hattenberger. Use of large-eddy simulations to design an adaptive sampling strategy to assess cumulus cloud heterogeneities by remotely piloted aircraft. *Atmospheric Measurement Techniques Discussions*, 2021:1–26, 2021.

[5] Sandrine Bony et al. EUREC4A: A Field Campaign to Elucidate the Couplings Between Clouds, Convection and Circulation. *Surveys in Geophysics*, 38(6):1529–1568, November 2017.

[6] B. Stevens et al. Eurec^{4}a. *Earth System Science Data Discussions*, 2021:1–78, 2021.

[7] Aurélien Cabarbaye, Titouan Verdu, Fabien Garcia, Michel Gorraz, Alexandre Bustico, Murat Bronz, and Gautier Hattenberger. Design of a high performance MAV for atmospheric research. In *IMAV 2019, 11th International Micro Air Vehicle Competition and Conference*, IMAV 2019, Madrid, Spain, September 2019.

[8] K. J. Sanchez, G. C. Roberts, R. Calmer, K. Nicoll, E. Hashimshoni, D. Rosenfeld, J. Ovadnevaite, J. Preissler, D. Ceburnis, C. O'Dowd, and L. M. Russell. Top-down and bottom-up aerosol–cloud closure: towards understanding sources of uncertainty in deriving cloud shortwave radiative flux. *Atmospheric Chemistry and Physics*, 17(16):9797–9814, 2017.

[9] R. Calmer, G. C. Roberts, K. J. Sanchez, J. Sciare, K. Sellegri, D. Picard, M. Vrekoussis, and M. Pikridas. Aerosol–cloud closure study on cloud optical properties using remotely piloted aircraft measurements during a bacchus field campaign in cyprus. *Atmospheric Chemistry and Physics*, 19(22):13989–14007, 2019.

[10] R. Calmer, G. C. Roberts, J. Preissler, K. J. Sanchez, S. Derrien, and C. O'Dowd. Vertical wind velocity measurements using a five-hole probe with remotely piloted aircraft to study aerosol–cloud interactions. *Atmospheric Measurement Techniques*, 11(5):2583–2599, 2018.

[11] G. de Boer, S. Borenstein, R. Calmer, C. Cox, M. Rhodes, C. Choate, J. Hamilton, J. Osborn, D. Lawrence, B. Argrow, and J. Intrieri. Measurements from the university of colorado raaven uncrewed aircraft system during atomic. *Earth System Science Data Discussions*, 2021:1–22, 2021.

[12] Pierre Crispel and Greg Roberts. All-sky photogrammetry techniques to georeference a cloud field. *Atmospheric Measurement Techniques*, 11(1):593–609, 2018.

Developing a modular tool to simulate regeneration power potential using orographic wind-hovering UAVs

M. Gossye, S. Hwang, and B.D.W. Remes*
Delft University of Technology, Kluyverweg 1, 2629HS Delft, the Netherlands

Abstract

Applications of Unmanned Aerial Vehicles (UAVs) are often limited by flight endurance. To address the limitation of endurance, we propose a regenerative soaring method in this paper. The atmospheric energy from updraft generated by obstacles such as hills and ships can be harvested by UAVs using a regenerative electric drivetrain. With fixed-wing aircraft, the vehicle can hover with specific wind condition, and the battery can be recharged in the air while wind hovering. In order to research the feasibility of this regenerative soaring method, we present a model to estimate hovering locations and the amount of extractable power using the proposed method. The resulting modular regeneration simulation tool can efficiently determine the possible hovering locations and provide an estimate of the power regeneration potential for each hovering location, given the UAVs aerodynamic characteristics and wind-field conditions.

1 Introduction

UAVs are performing more and more diverse missions every year, but are often limited by the maximum achievable endurance and/or range. Using the principle of orographic soaring to extend the range and endurance of UAVs has already been extensively researched, often based on techniques used by various bird species that have been observed [1, 2, 3]. However, conventional orographic soaring techniques do have some limitations that limit their usability in certain environments and conditions.

With traditional soaring method, the only energy-storage mediums are the potential energy (altitude) and kinetic energy (airspeed) of the aircraft. The associated aircraft state variables, altitude & airspeed, are often desired to stay constant to be able to take advantage of the favourable conditions to gain energy from the atmosphere [4]. A great example of this is when one is, for instance, soaring upwind along a ridge to try to take advantage of the updrafts it generates. It is possible to store the gained energy in the form of altitude, but the higher the altitude, the weaker the updrafts are from the obstacle. At a certain altitude, the updrafts become so weak that the glider is barely able to maintain altitude without losing airspeed. Once this ”ceiling altitude” is reached, it is not possible to store any more energy. It is possible to trade the potential energy for kinetic energy, and dive back down to the original altitude while gaining airspeed. The aircraft is now positioned once again in a region with stronger updrafts. However, due to the increased airspeed, the glider has a higher sink-speed which may render it unable to gain energy from the updrafts anymore.

Regenerative soaring introduces another energy storage medium to store harvested energy from the environment. The regenerative soaring method was first proposed by Paul MacCready already back in 1998 [5]. Instead of having to change the altitude and/or airspeed to be able to store energy, an onboard energy accumulator in the form of a rechargeable battery can harvest the energy through the use of a regenerative drivetrain. This means that the aircraft can stay positioned in the altitude region where the most favourable updrafts are present, and keep its optimum airspeed.

One problem with the suggested regenerative orographic soaring methods is that a long ridge or hill range is required to take advantage of this, such that the aircraft can fly straight along the ridge in the most favourable updraft regions for an extended period of time. It would be beneficial if small UAVs could also use the updrafts present around smaller, single objects such as a small hill, a building or a ship on the open sea. This could be achieved by altering the orographic regenerative soaring methods by applying a technique called wind hovering or static hovering.

Achieving static hovering while using the orographic soaring method (called wind hovering) is a topic that was found to only be covered by a very small amount of research.

Fisher [1] introduced the concept of a ”feasible soaring region”, a spatial region inside a wind-field where wind hovering is possible for a given wind-speed. A point in the wind-field is deemed feasible for wind hovering if the local vertical wind component/updraft velocity is larger or equal than the minimum sink speed of the aircraft when flying at zero ground speed in the wind field. In their paper, the feasibility of having a fixed-wing UAV autonomously hover in the updraft region of a hill and a building was investigated. The paper concluded with the experimental results proving that a small UAV can indeed apply wind-hovering techniques to statically hover in the favourable updraft region.

The remainder of this paper is structured as follows: Sec-

*Email address(es): midasgossye@gmail.com, S.Hwang-1@tudelft.nl, B.D.W.Remes@tudelft.nl tion 2 introduces the wind field estimation method, Section 3 describes how to calculate extractable power generated by the wind field, Section 4 presents how the feasible soaring locations and generated power at each location are determined. Finally, Section 5 gives a summary and further discussions.

2 WIND-FIELD ESTIMATION

To be able to determine the power available in the wind-field, it is first vital to have a good understanding of the wind-field. To achieve this, a wind-field estimation tool is required that can simulate the flow around various simple obstacles. The following subsections will describe what methods are available to achieve this and how the wind-field estimation program was implemented.

2.1 Methods

There exist numerous methods to estimate the behaviour of air around obstacles, greatly varying in complexity and required computational power. The most common choice lately has been to use a complex Computational Fluid Dynamics (CFD) simulation package like ANSYS fluent, openFOAM, etc. The CFD simulations performed with these packages require a large amount of computational power and are very complex to set-up. It was opted to first search for another method as a basis of the wind-field estimator. Langelaan used a simplified potential flow method to find the wind field upwind of an idealised circular shaped hill [6] to gain a better understanding of the general behaviour of the wind-field and to estimate the ideal location relative to the circular hill for ridge soaring. This methodology sparked the idea to use potential flow theory to estimate the flow field present upwind of the hill.

2.2 Potential flow estimator

The standard potential flow equations describing the idealised flow around circular and oval shaped obstructions were used as a basis.

The equations used to determine the flow-field are listed below, with U_∞ being the free-stream velocity, R the radius of the circular hill and r the distance between the aircraft and the centre of the hill. θ represents the angle between the horizontal axes and the radial of the aircraft:

$$u_r = \left[1 - \frac{R^2}{r^2}\right] U_\infty \cos\theta \tag{1}$$

$$u_\theta = -\left[1 + \frac{R^2}{r^2}\right] U_\infty \sin\theta \tag{2}$$

Transforming the polar velocity components into cartesian velocity components results in the following velocity functions for the x and y components:

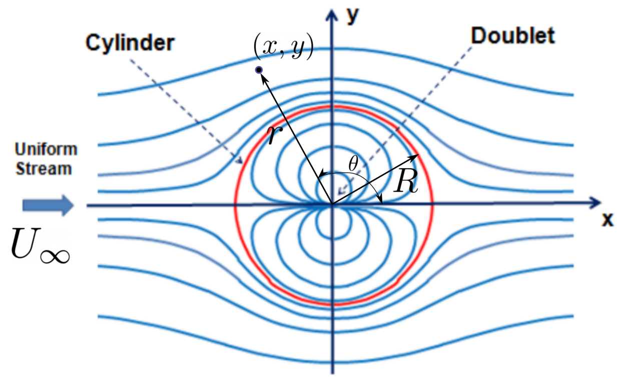


Figure 1: Potential flow field around cylinder

$$u_x = \cos\theta \cdot u_r - \sin\theta \cdot u_\theta \tag{3}$$

$$u_y = \sin\theta \cdot u_r + \cos\theta \cdot u_\theta \tag{4}$$

$$with \quad \theta = \arctan\frac{y}{x} \tag{5}$$

Another set of equations for oval shaped hills can also be selected, which correspond to the equations representing the flow-field over a rankine oval [7]:

$$x_{stag}^2 - a^2 - \frac{ma}{\pi U_\infty} = 0 \tag{6}$$

$$\Leftrightarrow m = \frac{\pi U_\infty}{a}(x_{stag}^2 - a^2) \tag{7}$$

$$u_x(x,y) = U_\infty + \frac{m}{2\pi}\left[\frac{x+a}{(x+a)^2+y^2} - \frac{x-a}{(x-a)^2+y^2}\right] \tag{8}$$

$$u_y(x,y) = \frac{my}{2\pi}\left[\frac{1}{(x+a)^2+y^2} - \frac{1}{(x-a)^2+y^2}\right] \tag{9}$$

Where the x-coordinate of the stagnation point x_{stag} and the x-coordinate of the focal point a determine the geometry of the oval shaped hill.

These equations were then altered with a simplified boundary/shear layer model equation to include an estimate of the Atmospheric Boundary Layer.

Equation 10 shows the used model that alters the vertical wind-speed distribution with a logarithmic function to try to estimate the Atmospheric Boundary Layer.

One problem arises by using this simple model to estimate the varying wind speeds in the boundary layer, the function is only able to estimate the boundary layer effects to the horizontal wind-speed over flat terrain. It has been proven though that the log wind-profile can produce accurate results even above non-flat terrain [8] in certain circumstances at higher altitudes above the obstacle. The log wall function can certainly be applied to the regions of the flow that are not greatly affected by the presence of the hill (mainly upwind of the hill-side). The proposed boundary layer model will however most likely not predict the boundary layer effects close to the hill surface. It was still opted to use this model for the entire hill region since the resulting flow patterns are more closely resembling real-life wind conditions where the flow velocity decreases close to the surface due to friction. If more accurate flow behaviour needs to be predicted close to the surface of the hill, a CFD simulation including models for laminar and turbulent boundary layer behaviour would be more applicable.

$$u(z_2) = u(z_1)\frac{\ln\left((z_2 - d)/z_0\right)}{\ln\left((z_1 - d)/z_0\right)} \tag{10}$$

3 Power contours

Now that a wind field estimate is available around differently sized obstacles, it is time to determine the feasible power that can be extracted at each point.

Before going into the details of the ability of the UAV to perform wind hovering at each location, it is helpful to first estimate the theoretical maximum power that can be extracted at each location assuming the UAV can maintain to hover at that location indefinitely. In this case, the energy harvesting UAV can essentially be modelled as a Horizontal Axis Wind Turbine (HAWT) where the upstream wind velocity is equal to the total wind velocity at the location of the UAV in the wind field. This is not totally accurate, since this assumes that the upstream wind velocity is constant along the axis of the propeller, but since the propeller dimensions of small UAV are at least an order of magnitude smaller than the obstacle dimensions it can be assumed that this will only have a very minor effect.

To be able to determine the theoretical maximum power that a HAWT can extract from the wind stream, it is evident to first have a closer look into the so called Betz law:

3.1 Betz law

One of the most famous theories concerning wind turbine theory is the Betz law (also called Betz condition or limit).

Simply put, it states that even an ideal wind turbine that contains no centre hub and has an infinite number of blades that cause no additional drag (e.g. skin friction drag) can only extract roughly 59 % of the power available in the wind stream [9]. For power to be continuously be able to be extracted, it is evident that a continuous mass flow of air must pass through the propeller/turbine disc. For this to occur, both the incoming and outgoing flow must have a positive flow velocity. If, hypothetically, the turbine was able to extract all of the available energy from the incoming flow, the flow past the disc area should have a velocity of zero (otherwise there would still be unextracted energy present). Having a zero fluid flow velocity at the exit of the turbine, directly means that no mass flow can be present, so no power can be extracted at these conditions.

Using the continuity equation, Euler's theorem and kinetic energy equations the following ideal power limit following the Betz law can be derived [9]:

$$P_{ideal} = \frac{16}{27}\frac{1}{2}\rho S_{turb} V_{air}^3 \tag{11}$$

This first estimate for the maximum theoretical power can be used as a basis to generate the power contours for the wind field. The following assumptions have to be kept in mind though:

- The wind turbine is assumed to not have a hub, the entire disc area region only contains blades
- It has an infinite number of blades that cause no additional drag (e.g. skin friction drag, induced drag due to tip vortices)
- The incoming flow is assumed to be constant, laminar and axial to the wind turbine axis
- No swirl is generated, the outgoing flow is also flowing axial to the wind turbine axis
- The air is considered to be an incompressible fluid

3.2 Using Betz law to generate potential power contours

Figure 2 shows the ideal maximum power at every location in the wind field that could be extracted from a $15\,\mathrm{m\,s^{-1}}$ free-stream velocity over a circular hill section with a radius of $50\,\mathrm{m}$ for a wind-turbine with a rotor disc area of $0.1\,\mathrm{m}^2$. It basically represents the absolute ideal maximum power that a regenerating UAV could achieve at every point in the wind field if static hovering can be achieved at that point and if the turbine can operate at its maximum power operating point, which will obviously not be the case for the majority of the wind field.

It is logical that the highest ideal power estimates are located directly above the hill since this is where the wind speeds are the highest (for the idealised potential flow case). It can be seen however that close to the surface of the hill the power figure is lower since this region has a lower velocity due to the added boundary layer wall function.

4 Determining hovering locations & power generation potential

Now that the absolute maximum theoretical power that can be extracted at each point in the wind field is known, the next step is to determine if the UAV can actually hover at that location, and if so, what power fraction should be extracted from the turbine to generate the required drag equalising the "thrust" generated by gravity, to enable the hovering to be stable?

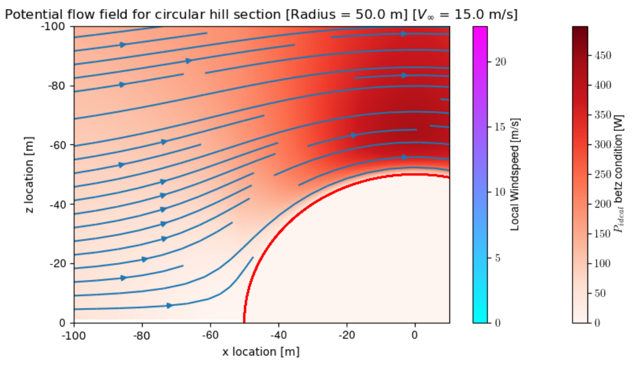


Figure 2: Ideal power contour plot for a $15\,\mathrm{m\,s^{-1}}$ free-stream velocity over a circular hill section with a radius of $50\,\mathrm{m}$ for a rotor disc area of $0.1\,\mathrm{m}^2$

To be able to answer this question, the equations governing the longitudinal flight dynamics of a hovering UAV needs to be studied.

4.1 *Longitudinal hovering flight dynamics*

The following equations (Equations 12, 13 and 14) express the system of differential equations for longitudinal flight dynamics (following from the FBD given in Figure 3), including a non-zero wind, in the air-path reference system [10]:

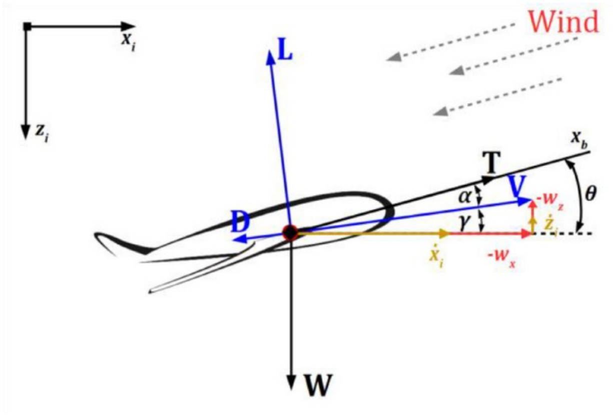


Figure 3: FBD Air path reference system longitudinal flight dynamics (courtesy of Langelaan [10])

$$T - D - W\sin\gamma = \frac{W}{g}\left(\dot{V}_{air} + \dot{u}_x\cos\gamma - \dot{u}_z\sin\gamma\right) \tag{12}$$

$$-L + W\cos\gamma = \frac{W}{g}\left(-\dot{V}_{air}\dot{\gamma} + \dot{u}_x\sin\gamma + \dot{u}_z\cos\gamma\right) \tag{13}$$

$$M = \ddot{\theta} I_{yy} \tag{14}$$

The equilibrium equations governing the balance of forces required for a UAV to hover in a steady state can be easily derived by setting the time derivative of the airspeed and both wind speed components (horizontal and vertical) to zero. The thrust force is also replaced with a (negative) turbine drag force which will represent the additional variable drag generated by propeller/motor drivetrain that can act as a turbine. To avoid possible confusions between the total drag force (encompassing both the aircraft and turbine drag forces) and the drag force purely generated due to the aerodynamic properties of the aircraft, the symbol D which represented the latter was replaced by D_{AC}. Lastly, it is assumed that all of the forces acting on the aircraft are acting at the CG, meaning no moments are generated. The simplifications and alterations are shown below in Equations 15 and 16. An altered FBD which reflects the changes and simplifications made is shown in Figure 4.

$$\cancelto{-D_{turb}}{T} - D_{AC} - W\sin\gamma = \frac{W}{g}\left(\cancelto{0}{\dot{V}_{air}} + \cancelto{0}{\dot{u}_x}\cos\gamma - \cancelto{0}{\dot{u}_z}\sin\gamma\right) \tag{15}$$

$$-L + W\cos\gamma = \frac{W}{g}\left(-\cancelto{0}{\dot{V}_{air}}\dot{\gamma} + \cancelto{0}{\dot{u}_x}\sin\gamma + \cancelto{0}{\dot{u}_z}\cos\gamma\right) \tag{16}$$

This results in the following system of equations:

$$\begin{cases} -D_{turb} - D_{AC} - W\sin\gamma & = 0 \\ -L + W\cos\gamma & = 0 \end{cases} \tag{17}$$

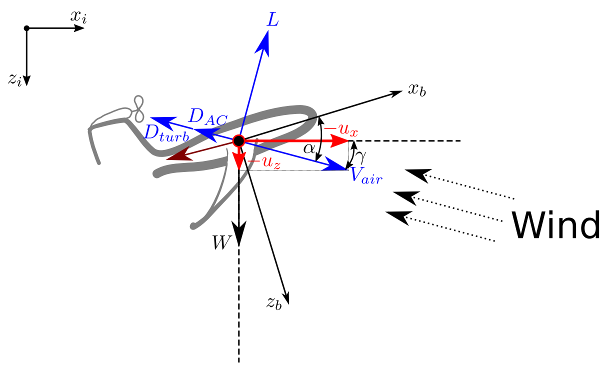


Figure 4: FBD Air path reference system longitudinal hovering flight dynamics

4.2 *Estimating turbine drag*

Assuming that the turbine behaves as an ideal wind turbine as discussed in Subsection 3.1, it can be assumed that the wind only exerts a net axial force on the rotor. This means the useful power that the wind turbine extracts can be written as the product of this axial force (D_{turb}) and the air velocity at the rotor disc/turbine (V_{turb}): $P_{turb} = D_{turb} \cdot V_{turb}$. Furthermore, when the rotor is operating at the theoretical maximum efficiency conditions Betz proved that the air velocity at the rotor disc/turbine must be equal to two thirds of the incoming air velocity [9]. Using these equations and observations, it is possible to derive a simple expression for the estimated drag produced by an ideal turbine which is shown below:

$$P_{ideal} = \frac{16}{27}\frac{1}{2}\rho S_{turb} V_{air}^3 \tag{18}$$

$$D_{turb} = \frac{P_{ideal}}{V_{turb}} \tag{19}$$

$$V_{turb} = \frac{2}{3} V_{air} \tag{20}$$

Substituting Equation 20 in Equation 19:

$$D_{turb} = \frac{P_{ideal}}{\frac{2}{3} V_{air}} \tag{21}$$

Finally. substituting Equation 21 in Equation 18 results in an equation expressing the estimated turbine drag (D_{turb}) in terms of incoming airspeed (V_{air}) and rotor disc area (S_{turb}):

$$\boxed{D_{turb} = \frac{1}{2}\frac{2}{9}\rho S_{turb} V_{air}^2} \tag{22}$$

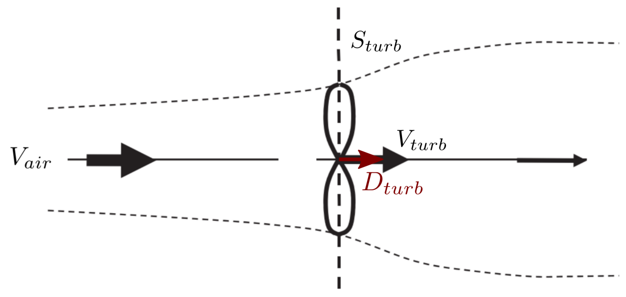


Figure 5: Ideal wind turbine diagram

4.3 *Finding the required lift and drag coefficients for hovering*

Now that both the systems of equations describing the force equilibrium during hovering flight and an estimate for the turbine drag are found, it is possible to derive a set of equations that determine the required lift and drag coefficients to enable static hovering.

Following from the system of equations that describes the force equilibrium during hovering flight derived in Subsection 4.1 (Equation 17), the required lift and drag terms can be expressed as follows:

$$\begin{cases} L & = W\cos\gamma \\ D_{turb} + D_{AC} & = -W\sin\gamma \end{cases} \tag{23}$$

Rewriting this system of equations in terms of the lift and drag coefficients results in the following system:

$$\begin{cases} \frac{1}{2}\rho V_{air}^2 S C_{L_{hover}} & = W\cos\gamma \\ \frac{1}{2}\frac{2}{9}\rho S_{turb} V_{air}^2 + \frac{1}{2}\rho V_{air}^2 S C_{D,AC_{hover}} & = -W\sin\gamma \end{cases} \tag{24}$$

Dividing both sides by $\frac{1}{2}\rho V_{air}^2 S$:

$$\begin{cases} C_{L_{hover}} & = \frac{W}{\frac{1}{2}\rho V_{air}^2 S}\cos\gamma \\ \frac{2}{9}\frac{S_{turb}}{S} + C_{D,AC_{hover}} & = -\frac{W}{\frac{1}{2}\rho V_{air}^2 S}\sin\gamma \end{cases} \tag{25}$$

The resulting non-dimensionalised contribution of the turbine to balance the horizontal force equilibrium (the bottom row of Equation 25), $\frac{2}{9}\frac{S_{turb}}{S}$, can be thought of being the maximum achievable drag coefficient of the turbine, since multiplying this figure by $\frac{1}{2}\rho V_{air}^2 S$ results in the ideal maximum drag caused by the turbine. Setting $C_{D_{turb}} = \frac{2}{9}\frac{S_{turb}}{S}$ results in the following system of equations:

$$\begin{cases} C_{L_{hover}} & = \frac{W}{\frac{1}{2}\rho V_{air}^2 S}\cos\gamma \\ C_{D_{turb}} + C_{D,AC_{hover}} & = -\frac{W}{\frac{1}{2}\rho V_{air}^2 S}\sin\gamma \end{cases} \tag{26}$$

Next, the sine and cosine of the flight path angle (γ) can be substituted with the fractions $\frac{u_z}{V_{air}}$ and $\frac{u_x}{V_{air}}$ respectively. This can be done because the velocity of the UAV with respect to the inertial reference frame is assumed to be zero during stable hovering. This means that the airspeed vectors magnitude and direction is purely determined by the local wind speed vectors (see Figure 4).

$$\begin{cases} C_{L_{hover}} & = \frac{W}{\frac{1}{2}\rho V_{air}^2 S}\frac{u_x}{V_{air}} \\ C_{D_{turb}} + C_{D,AC_{hover}} & = -\frac{W}{\frac{1}{2}\rho V_{air}^2 S}\frac{u_z}{V_{air}} \end{cases} \tag{27}$$

Finally, if the lift-drag polar can be estimated using the following standard equation relating the drag and lift coefficient to each other:

$$C_{D,AC} = C_{D_0} + \frac{C_L^2}{\pi A e} \tag{28}$$

And substituting this equation in Equation 27:

$$\begin{cases} C_{L_{hover}} & = \frac{W}{\frac{1}{2}\rho V_{air}^2 S}\frac{u_x}{V_{air}} \\ C_{D_{turb}} + C_{D_0} + \frac{C_{L_{hover}}^2}{\pi A e} & = -\frac{W}{\frac{1}{2}\rho V_{air}^2 S}\frac{u_z}{V_{air}} \end{cases} \tag{29}$$

This leaves a system of equations that can easily be solved for both the required lift coefficient ($C_{L_{hover}}$), and turbine drag coefficient $C_{D_{turb}}$ if the local air speed (which is equal to the wind speed magnitude during hovering), horizontal and vertical wind speed components are known.

Some important observations can be made from the final equations:

- The required lift coefficient, determined by the first part of Equation 29, should be less than the maximum lift coefficient of the aircraft. If this would not be the case, the aircraft would effectively stall when trying to achieve these conditions.

- The drag that the clean aircraft itself can provide is fixed by the required operating point on the lift-drag polar. If the required drag coefficient is lower than this value, the aircraft will not be able to achieve hovering equilibrium, even if the turbine is fully switched off or assumed to not be present;

- At specific wind speed and direction conditions, the clean aircraft will be able to provide just the right amount of drag at a certain required lift coefficient to satisfy both equilibrium equations, the turbine doesn't need to be switched on, and no power can be regenerated, since $C_{D_{turb}}$ will have to be equal to 0.

- At wind conditions where more drag is required than the clean aircraft itself can provide, the turbine needs to be switched on to close the "drag deficit" and equalise both terms of the second part of Equation 29. If the required extra drag from the turbine is less than its ideal maximum, the regen drivetrain should regulate the drawn power from the turbine in such a way that the drag provided by the turbine satisfies the equations.

- There exist another specific set of wind conditions where the required drag from the turbine to achieve hovering equilibrium will be equal to the maximum drag that the turbine ideally can provide. Note that although the maximum amount of power (imposed by the Betz limit) that can be drawn from the turbine in this scenario at the specific conditions, it is not necessarily the optimum resulting in the maximum amount of regeneration power, since the regeneration power also depends on the wind speed and other locations in the wind-field might exist where not all ideally available power can be extracted, but due to a higher wind speed the total regenerated power potential is still higher.

With the finalised equations for the turbine drag coefficient and above observations in mind, the calculation of the regen power contours can now be performed.

4.4 *Regen power contour calculation*

The finalised equations presented in the previous subsection enable one to determine if static hovering is achievable (given the local wind conditions at a certain point in the wind-field and aircraft parameters). If this is the case, the corresponding static hovering power regeneration potential can be calculated for that point.

The resulting equations can be used to determine both the required lift coefficient ($C_{L_{hover}}$), and combined drag coefficients (one being the turbine drag coefficient $C_{D_{turb}}$, the other being the drag coefficient of the aircraft $C_{D,AC_{hover}}$) to enable stable static hovering.

This function determines if the UAV is theoretically able to statically hover with zero ground speed at each point of the calculated wind field. At each potential hover location, the required additional drag and power needed from the turbine is calculated as well as the angle of attack.

First, the required lift coefficient to satisfy the hovering equilibrium equations is calculated:

$$C_{L_{hover}} = \frac{W}{0.5 \cdot \rho \cdot V_{air}^2 \cdot S} \cdot \frac{u_x}{V_{air}} \tag{30}$$

If the resulting lift coefficient is larger than the maximum achievable lift coefficient ($C_{L_{max}}$), the aircraft would stall if it tried to approach the conditions required for hovering and the corresponding point in the wind field will have a zero power regeneration potential using static hovering since hovering cannot be achieved.

Next, the total required drag coefficient to enable hovering ($C_{D_{turb}} + C_{D,AC_{hover}}$) is calculated:

$$C_{D_{turb}} + C_{D,AC_{hover}} = \frac{W}{0.5 \cdot \rho \cdot V_{air}^2 \cdot S} \cdot \frac{u_z}{V_{air}} \tag{31}$$

For the aircraft to be able to achieve static hovering, the combined required drag coefficient can not be smaller than the minimum achievable total drag coefficient. This minimum achievable total drag coefficient is equal to the clean aircraft's drag coefficient, since the least amount of drag will be generated when no additional turbine drag is generated (hence $C_{D_{min}} = C_{D,AC_{hover}} = C_{D_0} + \frac{C_{L_{hover}}^2}{\pi Ae}$).

The combined required drag coefficient can also not be larger than the maximum achievable drag coefficient, which is equal to the clean aircraft's drag coefficient plus the maximum achievable turbine drag coefficient. As stated in the previous subsection, the maximum achievable turbine drag coefficient can be estimated using the Betz limit and is equal to $C_{D_{turb,max}} = \frac{2}{9}\frac{S_{turb}}{S}$. Summarising, the acceptable combined required drag coefficient bounds to achieve static hovering leads to the following expression:

$$\begin{aligned} C_{D_0} + \frac{C_{L_{hover}}^2}{\pi Ae} &\leq C_{D_{turb}} + C_{D,AC_{hover}} \\ &\leq C_{D_0} + \frac{C_{L_{hover}}^2}{\pi Ae} + \frac{2}{9}\frac{S_{turb}}{S} \end{aligned} \tag{32}$$

If the total required drag coefficient falls within these bounds and the required lift coefficient is not larger than the maximum lift coefficient (as stated earlier), it can be assumed that the aircraft can achieve static hovering, and a valid power regeneration potential can be calculated.

The resulting required turbine drag coefficient to achieve stable static hovering can be calculated as follows:

$$C_{D_{turb,hov}} = C_{D_{turb}} + C_{D,AC_{hover}} - \left(C_{D_0} + \frac{C_{L_{hover}}^2}{\pi A e} \right) \tag{33}$$

The corresponding turbine drag generated during hovering can easily be found by multiplying the turbine drag coefficient with $0.5 \cdot \rho \cdot V_{air}^2 \cdot S$:

$$D_{turb,hov} = 0.5 \cdot \rho \cdot V_{air}^2 \cdot S \cdot C_{D_{turb,hov}} \tag{34}$$

Finally, by rearranging Equation 21 the estimated turbine power can be found:

$$P_{turb.hov} = \frac{2}{3} \cdot V_{air} \cdot D_{turb,hov} \tag{35}$$

4.5 Results

By incorporating the finalised turbine drag and power equations and only populating the values for locations where hovering is deemed feasible by satisfying the maximum lift coefficient constraint and conditions set in Equation 32, power contour plots can be generated for any given wind-field. This results in figures like the one shown below:

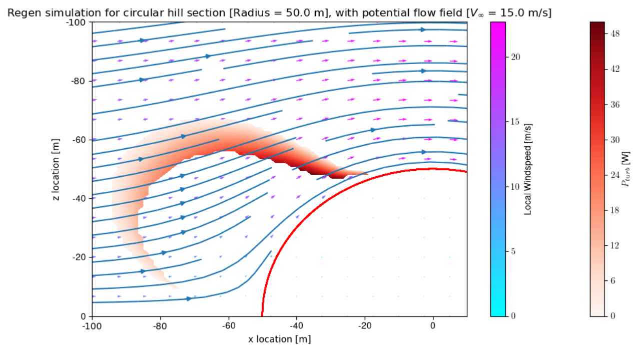


Figure 6: Regen power contour plot for a $15\,\mathrm{m\,s^{-1}}$ free-stream velocity over a circular hill section with a radius of $50\,\mathrm{m}$ for a rotor disc area of $0.1\,\mathrm{m}^2$ using aerodynamic parameters in Appendix A

It can immediately be seen that the estimated maximum amount of power that can be regenerated using the turbine while hovering is roughly 1 order of magnitude lower than then the ideal Betz limit power contour graph of the entire wind-field (see Figure 2). The primary reason for this is that the UAV is unable to statically hover with these conditions at the point in the wind-field that has the maximum potential power, which is the point with the highest wind velocity.

Power contour plots were calculated for a range of conditions, such as different wind-speeds, hill-sizes, rotor disc areas, UAV masses, etc. The resulting plots showed the expected behaviour for the change in conditions.

5 Conclusion

A simplified wind-field model around obstacles such as circular and oval shaped hills was constructed based on potential flow theory. A model was developed to determine the maximum theoretical regeneration power if wind-hovering orographic soaring techniques are applied, given a certain wind-field and aerodynamic characteristics of the UAV. The resulting modular simulator program is able to determine the hovering locations and gives an estimate of the maximum achievable regeneration power. For the source wind-field, either the simplified potential-flow based model can be used, which needs very little computational power making it suitable to be even run on on-board processors of UAVs, or a wind-field generated by other more advanced software or even from a measurement field. The tool should allow anyone to easily get an estimate of the feasibility of the regenerative hovering soaring method in their particular application.

For future work, additional simulations could be carried out by simulating real-life conditions. The model can then be validated by performing a flight-test in these conditions with a regenerative drivetrain architecture like the one being proposed and shown in Appendix B.

Acknowledgements

We would like to thank our colleagues from the Never-Landing Drone working group for their continued support, suggestions, ideas and feedback.

References

[1] Alex Fisher, Matthew Marino, Reece Clothier, Simon Watkins, Liam Peters, and Jennifer L. Palmer. Emulating avian orographic soaring with a small autonomous glider. *Bioinspiration and Biomimetics*, 11(1), 12 2015.

[2] Nicholas R.J. Lawrance and Salah Sukkarieh. Wind energy based path planning for a small gliding unmanned aerial vehicle. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2009.

[3] N R J Lawrance, J J Acevedo, J J Chung, J L Nguyen, D Wilson, and S Sukkarieh. Long Endurance Autonomous Flight for Unmanned Aerial Vehicles Aerial Robotics Long Endurance Autonomous Flight for Unmanned Aerial Vehicles. *AerospaceLab*, (8), 2014.

[4] Ramiro Carvalho. Development of the regenerative soarer: Theoretical and Practical aspects. In *31st Congress of the International Council of the Aeronautical Sciences*, number September, Belo Horizonte, 2018.

[5] Paul MacCready. Regenerative battery-augmented soaring. In *Self-Launching Sailplane Symposium*, 1998.

[6] Jack W. Langelaan. Long distance/duration trajectory optimization for small UAVs. *Collection of Technical Papers - AIAA Guidance, Navigation, and Control Conference 2007*, 4(August):3654–3667, 2007.

[7] A A Sonin. 2 .25 Advanced Fluid Mechanics. (c):2–4, 2011.

[8] S. Besio, A. Mazzino, and C. F. Ratto. Local log-law-of-the-wall in neutrally-stratified boundary-layer flows. *Boundary-Layer Meteorology*, 107(1):115–142, 2003.

[9] Magdi Ragheb and Adam M. Ragheb. Wind Turbines Theory - The Betz Equation and Optimal Rotor Tip Speed Ratio. In Rupp Carriveau, editor, *Fundamental and Advanced Topics in Wind Power*, page 21. Intech, 2011.

[10] N. Gavrilovic, A. Mohamed, M. Marino, S. Watkins, J. M. Moschetta, and E. Benard. Avian-inspired energy-harvesting from atmospheric phenomena for small UAVs. *Bioinspiration and Biomimetics*, 14(1), 2019.

APPENDIX A: UAV AERODYNAMIC PARAMETERS USED FOR SIMULATION

S	1	m^2
C_{L_α}	5.7	rad^{-1}
α_{0L}	-4	°
A	6	-
e	0.8	-
C_{D_0}	0.05	-

Table 1: UAV aerodynamic parameters for simulation

APPENDIX B: PROPOSED REGEN ARCHITECTURE

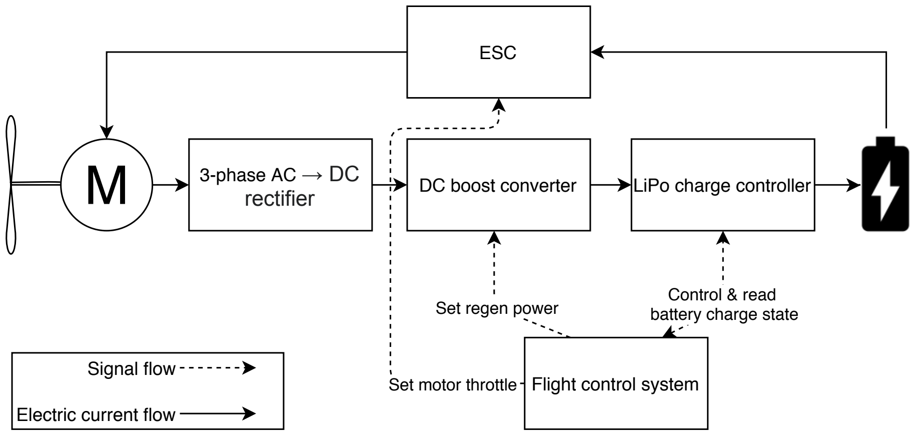


Figure 7: Example regen architecture

Estimating wind using a quadrotor

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

ABSTRACT

The aim of this work is to estimate the wind that the quadrotor drone is subject to only based on standard navigation sensors and equations of motion. It can be used in several situation, including atmospheric studies, trajectory planning under environmental constraints, or as a reference for studying flights in shear layer. For this purpose, a small quadrotor drone with spherical shape has been developed. Flight data are recorded from telemetry during indoor and outdoor flight tests and are post-processed. The proposed solution is based on a calibration procedure with global optimization to extract the drag model and a Kalman Filter for online estimation of the wind speed and direction.

1 INTRODUCTION

Estimating wind with a UAV has already been studied with multiple approaches. A common way is to use a fixed-wind aircraft and extract the wind from its GPS track [1] or by adding sensors on the system, such as 5-hole probe [2, 3] or a combination of simple Pitot tube and flags [4].

For a quadrotor, these approaches are not suitable, not only due to the non-constant inclination angles and flight directions, but also because of the low flight speeds. However, [5] compared the use of four different anemometers on a quadrotor. The study revealed that a thermal anemometer could be used, at the cost of modifications to the UAV structure in order to place it far enough from the disturbances induced by propellers.

Instead of adding components on the UAV, an alternate solution is to estimate wind from the quadrotor motion [6, 7, 8, 9]. In [7], the different possible models are presented: static, kinematic or full dynamic. In addition, a methodology to extract the required parameters is presented. The propulsion system is characterized by a motor test bench in a wind tunnel experiment, while the drag is extracted from flights at constant GPS velocity in steady air. The observation is made that the drag is proportional to the relative airspeed. In [6, 10], experiments were led using a six-axis force balance, a very precise but fragile and expensive system. Finally, [8] presents a nonlinear observer able to accurately predict the wind components, using only low cost Inertial Measurement Unit (IMU) and ground speed measurements. The drag-force is considered proportional to the rotational speed of the motors, that is almost constant during operation, leading to a constant rotor drag coefficient, similar to [7, 11]. [9, 7] have performed outdoor flights and compared the results with ground reference measurements, demonstrating the feasibility of wind measurement from quarotor based on IMU and GPS measurements. The general principals and equations of motion from these studies have been used as a starting point for the present article.

This paper is organized as follows. First, the problem modeling focuses on the equations of motion, the hypothesis and limitations, as well as the experimental airframe. Then, the parameters identification method is presented and after that, the wind estimation with a Kalman filter is exposed. Finally, in-flight experiments are described and their results are analyzed.

2 PROBLEM FORMULATION

2.1 *Kinematic and Dynamic model*

With the assumptions that

- the center of gravity (CG) is located at the center and origin of the body frame
- the frame and the propellers are rigid
- the inputs of the system are the thrust generated by the motors
- the outputs are the position and orientation of the body frame relative to the earth (inertial) frame, observed by the GPS and IMU sensors

a simple dynamic particle model can be established as in [7], leading to:

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

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

where:

- $\mathbf{X}$ is the position vector relative to earth frame
- $\mathbf{V_k}$ is the ground speed vector relative to earth frame (inertial velocity)
- $\mathbf{V_r}$ is the relative air speed vector
- $\mathbf{V_w}$ is the wind speed vector relative to earth frame
- equation 1 represents the wind triangle
- m is the mass of the model and $\mathbf{g}$ the gravity vector

- $V_a = \|\mathbf{V_r}\|$ is the norm of the airspeed
- **D** is the drag vector in earth frame, as a function of airspeed
- **T** is the control forces vector (thrust) in earth frame

Finally, the last assumption is that the wind speed is seen as a constant or slowly varying disturbance, therefore the derivative of the wind triangle (equation 1) gives:

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

A typical drag equation is expressed as the product of the dynamic pressure, a reference surface and a drag coefficient function of α and β, the the two angles defining the direction of the air velocity relative to the body frame (see Figure 1). As a result, drag should be proportional to V_a^2. However,

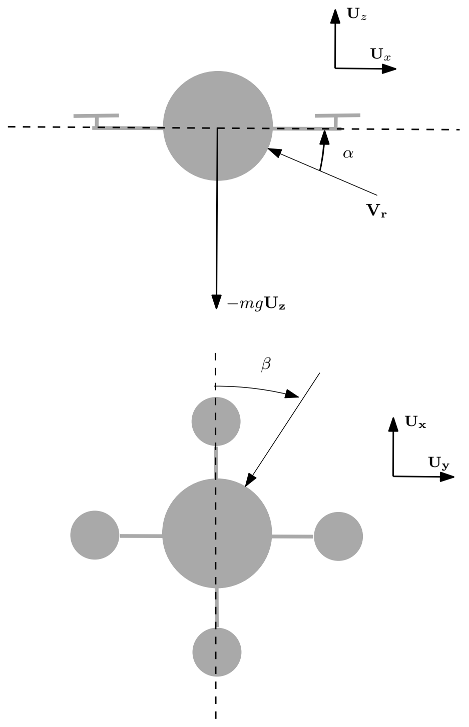


Figure 1: Aerodynamic frame with angle of attack α and side slip angle β

the experimental results presented in the calibration section 3 show that in the range of the considered wind speeds, the rotor drag or H-force (see [11]) that is linear with the airflow seems to be dominant, hence:

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

As an additional remark, the airframe is not not supposed to generate lift and is symmetrical to reduce the dependency with the direction of the relative airspeed (independent of β).

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

$$\mathbf{T} = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix} = \mathbf{R_{0b}}^{-1} \begin{pmatrix} 0 \\ 0 \\ T_{total} \end{pmatrix} \tag{5}$$

2.2 *Airframe characteristics*

A custom quadrotor frame have been designed for this experiment. It is a simple cross shape made of thin aluminum bars to hold the motors and a spherical 3D-printed central body around the electronic components and the battery, as seen on the Figure 2. The reason for this choice is to have a symmetrical shape in order to reduce or eliminate the dependency between the generated drag and the heading of the drone.

The general characteristics and components are summarized in the Table 1. The autopilot software used is the *Paparazzi* UAV System [12].

component	characteristic
material	aluminum & plastic (PLA)
motors	T-motor F30
propellers	Dalprop 5x4 (3 blades)
battery	3S, 2200 mAh
autopilot	Tawaki v1 with Paparazzi
GPS	U-blox M8
size (motor to motor)	47 cm
sphere diameter	22 cm
mass	896 grams
flight time	7 minutes

Table 1: Quadrotor characteristics

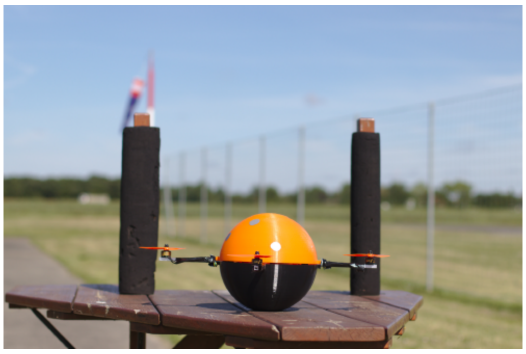


Figure 2: Quadrotor with custom spherical body shape

3 Calibration methodology

Estimating the wind from the quadrotor motion requires to estimate the drag parameters. The methodology applied is similar to [7] with the measurement of the bank angles ϕ and θ at different airspeed. One of the difference is that instead of moving at constant ground speed in steady air, the drone is controlled to stay at the same position in front a *WindShape* wind generator, as seen Figure 3. The calibration is done inside a flight arena at ENAC (École Nationale de l'Aviation Civile, Toulouse, France) also equipped with an *Optitrack* motion capture system. Hence the position and velocity is accurately controlled in closed-loop and the flight can be considered in equilibrium.

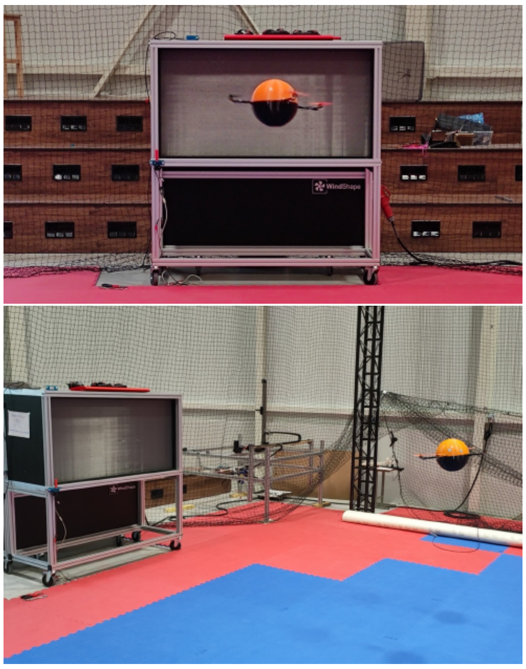


Figure 3: Quadrotor during calibration in from of the WindShape wind generator

The dynamic equation 2 in equilibrium state, illustrated by the Figure 4, directly provides a measurement of the drag force from the bank angle and the mass. Without loss of generality, the vertical component of the thrust compensate the weight and the horizontal thrust compensate the drag:

$$\begin{cases} T_{total}\ cos(\alpha) = T_z = mg \\ T_{total}\ sin(\alpha) = T_x = D_x = \|\mathbf{D}\| \end{cases} \Leftrightarrow mg\ tan(\alpha) = D_x$$

The calibration procedure is as follow:

- start the wind generator and measure the reference wind speed with an anemometer (hot-wire in our case)
- takeoff and place the quadrotor at a distance corresponding to the reference measurement

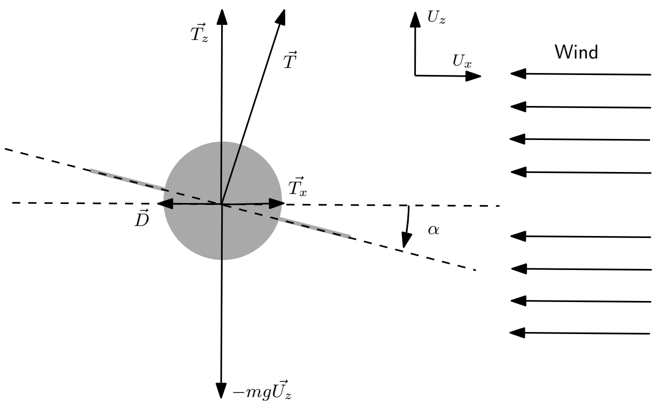


Figure 4: Side view of the forces applied on the quadrotor model in presence of horizontal wind

- when stabilized, change the heading to make one or more full turn on itself
- record the attitude (ϕ, θ, ψ)
- repeat the procedure at a different wind speed

For each reference wind speed, a fitting algorithm is used to map the relation between the bank angles (ϕ or θ) and the heading ψ with a sinusoidal form:

$$\phi \text{ or } \theta = c_1\ sin(c_2\ \psi + c_3) \tag{6}$$

where the coefficient c_1 is the magnitude of the oscillation, equal to the incidence α_{eq} at equilibrium state, c_2 and c_3 are frequency and phase parameters, not used later on. The Figure 5 shows the bank angle ϕ as a function of the heading ψ after a full turn. The rotation is not continuous, but made a 3s steps every after a rotation of 20°. The reference speed for this case is 6.57 m/s, corresponding to a 50% throttle of the wind generator. The curve fitting is done with Matlab and gives the result $c_1 = 0.1274$ with a R-square of 0.9271.

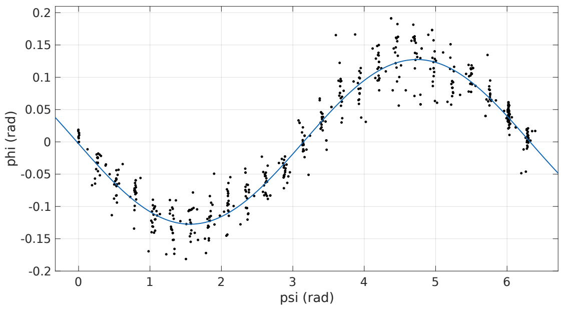


Figure 5: Roll angle ϕ against heading angle ψ after a complete turn in from of the wind generator, the sinusoidal fitting curve is in blue

Similar results are obtained for θ, which confirms this independence of the drag with β, and for the different reference speeds. The final result is presented with the Figure 6 that represents the relation between the angle of attack α_{eq} and the reference air speed. As mentioned in the previous section with equation 4, the relation is linear and the fitting curves gives:

$$tan(\alpha_{eq}) = c_\alpha * V_a$$
$$c_\alpha = 0.0262$$

with a R-square of 0.9998. It gives for our quad model with a mass of 896 grams a coefficient $k = mg * c_\alpha = 0.230$ in equation 4.

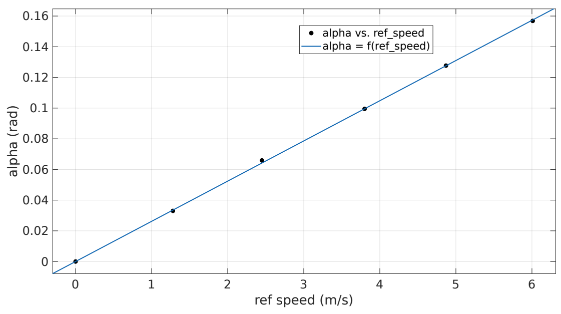


Figure 6: Relation between the incident angle α versus the reference speed measured from a hot-wire airspeed probe at the same distance from the wind generator

4 Wind estimation with Kalman filter

This problem considers simultaneous estimation of $2D$ aircraft Earth-axis velocity $\mathbf{V_r} = (V_{r_x}, V_{r_x})$ and wind velocity components (V_{w_x}, V_{w_y}). Both process and measurement equations are dependent on aerodynamic force described above. The attitude and heading estimation (AHRS filter) is performed through a fusion algorithm of low-cost inertial sensors used for UAV navigation.

All the sensors embedded are low-cost, and therefore have imperfections. The major error sources in the navigation system are due to: - all of the disturbances (noises) that affect the instruments; - the potential incorrect navigation system initialization (e.g. on magnetometers sensor); - and the inadequacy between the real local Earth's gravity value and the one used for computation. The largest error is usually a bias instability (expressed respectively in deg/hr for gyros and μg for the accelerometers). All these measurements are obviously corrupted by additive noises for which it appears reasonable to assimilate their stochastic properties to the ones of Gaussian processes. Their covariances matrices have been identified in [13].

Using these values, the state space representation corresponding to $\mathscr{M}_s$ can be described in a linear form:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} + \nu \qquad \text{and} \qquad \mathbf{y} = C\mathbf{x} + \mu$$

where: $\mathbf{x} = [V_{r_x}, V_{w_x}, V_{r_y}, V_{w_y}]^T$, $\mathbf{u} = [T_x, T_y]$ and $\mathbf{y} = [V_{k_x}, V_{k_y}]^T$ are the state, input and output vectors respectively. Moreover, ν, μ are the zero-mean Gaussian process noise vectors with covariance matrix, Q, R.

$$\mathscr{M}_s \begin{cases} \dot{\mathbf{x}} = \begin{pmatrix} \frac{-k}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-k}{m} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \mathbf{x} + \begin{pmatrix} \frac{1}{m} & 0 \\ 0 & 0 \\ 0 & \frac{1}{m} \\ 0 & 0 \end{pmatrix} \mathbf{u} + \nu \\ \begin{pmatrix} V_{k_x} \\ V_{k_y} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \mathbf{x} + \mu \end{cases}$$

Obviously, to implement these equation a discrete form is used such that :

$$\begin{cases} \mathbf{x}_{k+1} = A_d \mathbf{x}_k + B_d \mathbf{u}_{k-1} + \nu_k \\ y_k = C_d \mathbf{x}_k + \mu_k \end{cases}$$

where $A_d = \exp(Adt)$,$B_d = \int_0^{dt} \exp(A\xi)Bd\xi$,$C_d = C$ are the discrete-time state matrices and dt is the sampling time of the system. Process and measurement equations becomes :

$$\mathscr{M}_d \begin{cases} \begin{pmatrix} V_{r_x} \\ V_{w_x} \\ V_{r_y} \\ V_{w_y} \end{pmatrix}_{k+1} = \begin{pmatrix} 1 - \frac{kdt}{m} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 - \frac{kdt}{m} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{x}_k + \dots \\ \qquad \begin{pmatrix} \frac{dt}{m} - \frac{kdt^2}{2m^2} & 0 \\ 0 & 0 \\ 0 & \frac{dt}{m} - \frac{kdt^2}{2m^2} \\ 0 & 0 \end{pmatrix} \mathbf{u}_{k-1} + \nu_k \\ \begin{pmatrix} V_{k_x} \\ V_{k_y} \end{pmatrix}_k = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \mathbf{x}_k + \mu_k \end{cases}$$

Computing the rank of the observability matrix shows that the system is fully observable at all speed. It can be mentioned this results from the linear relation of drag with airspeed in equation 4. The case of a sphere without propellers in a airflow would give a drag as the square of the airspeed and a loss of observability at the zero airspeed, in addition with the need of an Extended Kalman filter form.

In the later experiments, the process noise relative to airspeed evolution is $\nu_{V_a} = 0.05$, the process noise relative to windspeed evolution is $\nu_{V_w} = 0.001$ and the sensor noise associated to ground speed measurement is $\mu_{V_k} = 0.1$. Second order terms (in dt^2) can be neglected.

5 Experimental flights

5.1 Indoor flight in controlled wind speed

The wind estimation algorithm is first evaluated with a flight indoor in front of the wind generator used for calibration. The goal is to evaluate the stability and convergence of the estimation in a controlled environment. In this setup, the wind is coming from a virtual north and the quadrotor is hovering at a fixed position. Ground speed and orientation are recorded from the telemetry and post-process by the Kalman filter in a Matlab script.

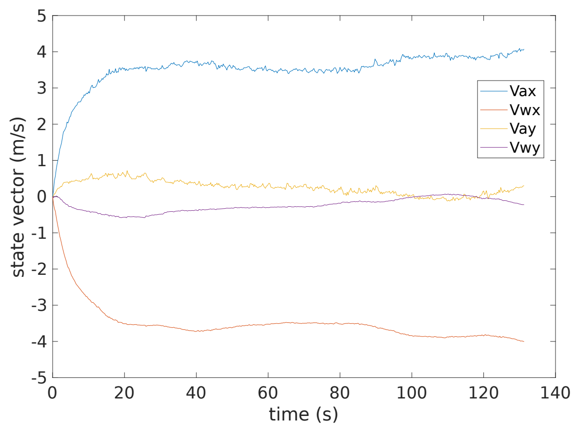


Figure 7: Evolution of the Kalman filter state vector over time during indoor flight

The Figure 7 shows the four elements of the state vector. The convergence time is around 10 to 20 seconds. The noise on the estimated airspeed is low thanks to the low noise on the measured ground speed (from motion capture) and the stable airflow.

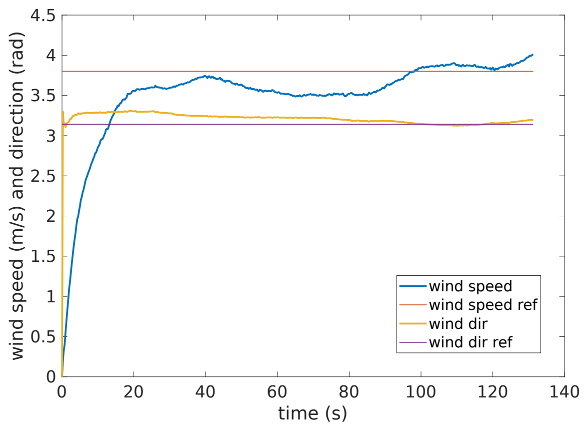


Figure 8: Wind speed and direction compared to the reference value from the wind generator; wind is coming from a virtual north (π rad) at a speed of 3.8 m/s

The Figure 8 is the norm of the estimated wind speed $\|\mathbf{V_w}\| = \sqrt{V_{w_x}^2 + V_{w_y}^2}$ and the wind direction. During this experiment, the wind generator throttle was set to 40%, corresponding to 3.8 m/s at the location of hovering with a direction of 180°. As we can see on this plot, the estimated wind converges to the reference values. The response time for the direction is much faster than for the norm of the speed. This can be explained by the fact that our model states that the wind has a very slow evolution. Since the initial condition for the state vector is zero, it takes some time to reach the final value, but even if the norm is not yet correct, the direction of the wind is already valid.

5.2 *Outdoor flights*

Several flights have been performed outdoor to record telemetry data, with attitude from IMU and this time ground speed from real GPS sensor. Only two relevant flights are presented in this paper. They were performed at Muret's model airfield (close to Toulouse, France) on the 20th of May 2021. The wind conditions for that day are coming from public meteorological data and are reported in Table 2.

time	speed	direction
9 am	no wind	N/A
10 am	7.4 km/h (2.06 m/s)	110°
11 am	9.3 km/h (2.58 m/s)	120°
12 am	9.3 km/h (2.58 m/s)	70°

Table 2: Wind condition on the day of the outdoor experiment

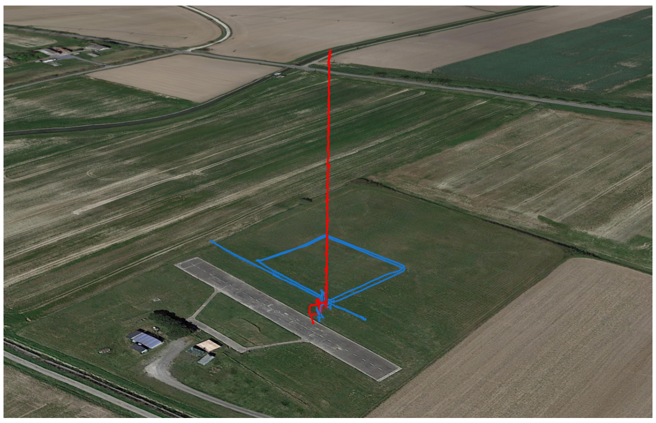


Figure 9: Trajectories for the outdoor flights: vertical profile in red, horizontal square at constant altitude in blue

The first flight corresponds to the vertical profile (red trajectory on Figure 9). The quadrotor goes up and down at a vertical speed of 1 m/s and with the heading changing at constant rate. The horizontal speed is close to zero. The state vector evolution is plotted on Figure 10. As expected, the noise on airspeed estimate is stronger than during indoor experiment due to sensor noise, but still acceptable.

The vertical flight was done a bit after 10 am, so according to Table 2 between 2 and 2.5 m/s, with a direction between 110° and 120°. The speed of 2.2 m/s and the direction of 115° have been kept as a reference. The Figure 11 shows

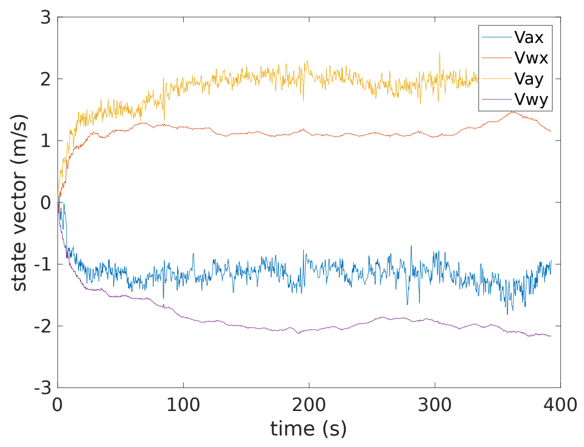


Figure 10: Evolution of the Kalman filter state vector over time during outdoor flight (vertical profile)

the norm and direction of the estimated wind compared to these reference values. It can be seen that both values are converging smoothly to the expected values.

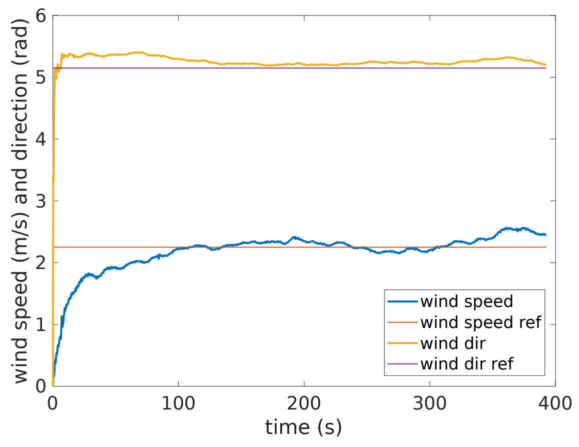


Figure 11: Wind speed and direction compared to the reference value from public meteo data during the outdoor flight (vertical profile)

The second flight is a square (blue trajectory on Figure 9) at low altitude. As a consequence, the ground speed is changing, with horizontal acceleration when changing direction. These variations can be seen in the airspeed vector on Figure 12. The wind vector is also showing some variations and is less stable than with the previous case.

The square flight was done at 9:30, so according to Table 2 between 0 and 2 m/s, with a direction around 110°. The reference speed of 1 m/s have been kept. The norm and direction are still converging, but with more variations due to the changes in ground speed as mentioned before. Nevertheless, all parameters stay in acceptable range and provide a valid estimation of the wind components.

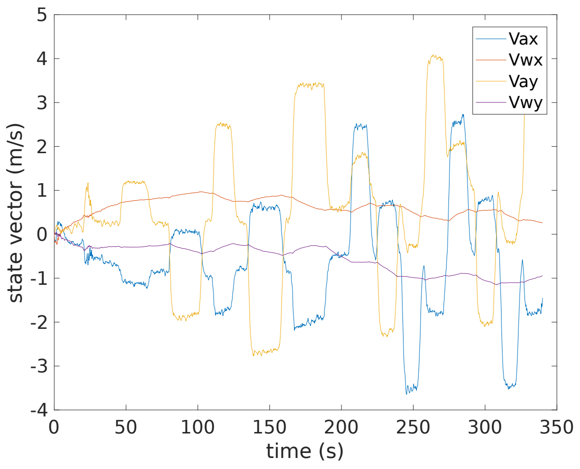


Figure 12: Evolution of the Kalman filter state vector over time during outdoor flight (square trajectory)

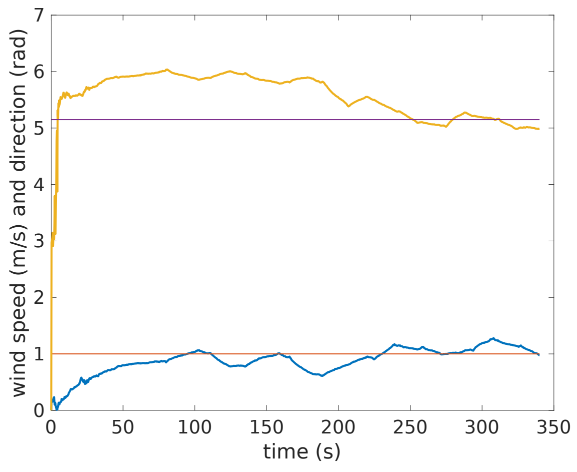


Figure 13: Wind speed and direction compared to the reference value from public meteo data during the outdoor flight (square trajectory)

6 CONCLUSION

A wind estimation filter based on the orientation and ground speed of a quadrotor have been presented. Experimental flights were conducted to compute the calibration parameters of the model and validate the results in controlled indoor flight, as well as outdoor conditions.

The next step is the implementation of the Kalman filter in the flight controller of the autopilot to provide real-time estimation on-board. Such information can be useful to improve navigation or even state estimation.

ACKNOWLEDGEMENTS

This work have been done using the preliminary work and calibration experiments of Laurent Hublet, Maxime Vacheron and Carl Guilly, students at Enac, during their second year project under the supervision of the authors.

REFERENCES

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

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

[3] S. Prudden, A. Fisher, M. Marino, A. Mohamed, S. Watkins, and G. Wild. Measuring wind with small unmanned aircraft systems. *Journal of Wind Engineering and Industrial Aerodynamics*, 176:197–210, 2018.

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

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

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

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

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

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

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

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

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

[13] Murat Bronz, Jean-Philippe Condomines, and Gautier Hattenberger. Development of an 18cm Micro Air Vehicle : QUARK. In *IMAV 2013, International Micro Air Vehicle Conference and Flight Competition*, Toulouse, France, September 2013.

Hybrid UAV Attitude Control using INDI and Dynamic Tilt-Twist

Lars F.A. Dellemann and Christophe De Wagter*
Delft University of Technology, Micro Air Vehicle Lab, Kluyverweg 1, 2629HS Delft, the Netherlands

ABSTRACT

The increased search for the performance of Unmanned Aerial Vehicles (UAVs) has led to an interest in hybrid concepts like the tail-sitter UAV. A tail-sitter UAV is capable of combining vertical take-offs and landings (VTOL) with efficient long-endurance forward flights. During hover, the wings do not provide lift but instead act as disturbance and limit the yaw response. Attitude control based on direct quaternion feedback does not take the differences in reaction speed for the three axes into account. Tilt-twist control has been proposed to overcome this problem as it splits the faster tilt (pitch and roll) from the slower and less important twist (yaw) and is successfully applied to quadrotor control. This paper proposes a novel tilt-twist controller based on Incremental Nonlinear Dynamic Inversion (INDI). But in tail-sitter UAVs, the lift vector can differ a lot from the tilt angle, especially when partly or fully transitioned to forward flight. To address this, a dynamic tilt-twist controller is proposed that redefines the twist according to the transition angle. Simulations and test flight tests are performed with the NederDrone hybrid tail-sitter to show the increased performance.

1 INTRODUCTION

The market for unmanned aerial vehicles is increasing [1]. Due to the development of small processors, UAVs became widely available for the public [2]. Many companies are currently developing UAVs for various purposes [3] [4]. The reasons to use this type of aircraft are their low cost, high maneuverability, and ease of use. Quadcopters can perform VTOL but have limited flight endurance. Fixed-wing UAVs can cover larger distances than quadcopters but are not able to hover. A solution to achieve both is to use a hybrid UAV (Figure 1).

The most common hybrid UAV types are tilt-rotors [5], tilt-wings [6], tail-sitters [7] and quadplanes [8]. Both the tilt-rotors and tilt-wings have components that can rotate during the transition between hover and forward flight. A quadplane uses different actuators for hover and forward flight. A tail-sitter uses the same actuators in both flight phases and rotates the entire body of the aircraft. One of the challenges with this type of UAV is the controllability during the landing phase, especially in turbulence [9]. During hover, some of the actuators become less efficient while the wings create important perturbing forces in turbulent or hard wind.

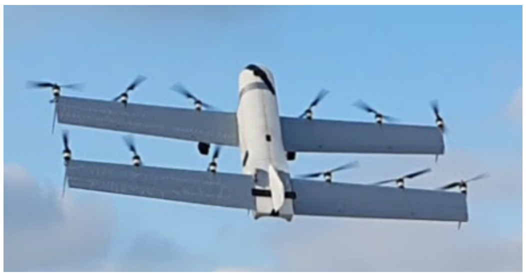


Figure 1: The NederDrone, capable of performing VTOL. The tail-sitter has 20 actuators (12 motors and 8 elevons) and the energy is stored in a hydrogen tank.

1.1 *NederDrone*

The tail-sitter used for this project is the NederDrone [7], shown in Figure 1. The NederDrone has two wings that carry twelve engines and hold eight elevons. The twelve engines are all used during hover, but only four are used during forward flight. One characteristic is that this UAV can achieve much larger roll moments than pitch moments, and even smaller yaw moments while the yaw also experiences a lot of aerodynamic damping from the wings.

Simple quaternion control finds the shortest rotation between the current attitude and desired attitude. It then assumes that all three axes can perform the desired rotation at the same time. In hybrid aircraft like the Nederdrone where not every axis is as fast, this results in either undesired rotations or slowing down the fast axes to match the slowest. Both options are undesirable. Tilt-twist control was proposed to address this problem by splitting the tilt error (controlled by the faster pitch en roll rotations) and the twist error (controlled by the slower yaw axis) [10]. But this work used a classical controller.

Recent advances in control have shown the benefits of sensor-based approaches like Incremental Non-linear Dynamic Inversion (INDI) [11]. But INDI assumes that all rotations can be executed at the same time and that the control

*Email address(es): c.dewagter@tudelft.nl demand can be met by the actuators. In hybrid aircraft such as the Nederdrone, which fly in harsh weather conditions, this is not the case.

This paper, therefore, proposes a combination of the tilt-twist method and INDI control, called the dynamic tilt-twist.

In Section 2 the theory behind the feedback error is given for quaternion, tilt-twist, and dynamic tilt-twist. Section 3 describes the methodology and results for the simulation and real-life test. Conclusions are drawn in Section 4.

2 METHOD

2.1 Axis definition

An axis system uses a body-fixed system, where the z-axis is parallel to the gravitational force during hover when the pitch and roll angles are zero. The x-axis goes through the belly of the UAV and the y-axis through the right wing (see Figure 2). In forward flight, the x-axis thereby becomes perpendicular to the gravitational force.

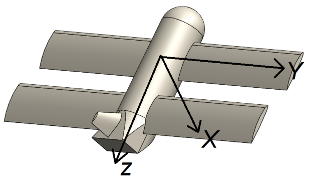


Figure 2: Body-fixed axis definition. If the pitch and roll angles are 0 degrees, the z-axis is parallel to the gravitational force vector.

2.2 Quaternion

The quaternions describe the transition of the attitude in one single rotation [12], with a rotation η and a three-dimensional unit vector component $\boldsymbol{r}$ as in

$$\boldsymbol{q} = \begin{bmatrix} \cos(\frac{\eta}{2}) \\ \boldsymbol{r}\sin(\frac{\eta}{2}) \end{bmatrix} = \begin{bmatrix} q_i & q_x & q_y & q_z \end{bmatrix}^\top \quad (1)$$

The current attitude is defined as

$$\boldsymbol{q}_c = \begin{bmatrix} q_{ci} \\ q_{cx} \\ q_{cy} \\ q_{cz} \end{bmatrix} \quad (2)$$

and the desired attitude (input) is

$$\boldsymbol{q}_d = \begin{bmatrix} q_{di} \\ q_{dx} \\ q_{dy} \\ q_{dz} \end{bmatrix} \quad (3)$$

The attitude error then becomes

$$\boldsymbol{q}_{err} = \boldsymbol{q}_d \otimes \boldsymbol{q}_c^{-1} \quad (4)$$

$$\boldsymbol{q}_{err} = \begin{bmatrix} q_{err1} \\ q_{err2} \\ q_{err3} \\ q_{err4} \end{bmatrix} = \begin{bmatrix} q_{d0} & q_{d1} & q_{d2} & q_{d3} \\ -q_{d1} & q_{d0} & q_{d3} & -q_{d2} \\ -q_{d2} & -q_{d3} & q_{d0} & q_{d1} \\ -q_{d3} & q_{d2} & q_{d1} & q_{d0} \end{bmatrix} \boldsymbol{q}_c \quad (5)$$

The controller then sends the errors in x-, y- and z-axis to the respective actuators through a PD reference generator. δ_a, δ_e and δ_r represent the aileron deflection, elevator deflection and rudder deflection respectively.

$$\delta_a = -2(k_p q_{err_2} + k_d \dot{q}_{err_2}) \quad (6)$$

$$\delta_e = -2(k_p q_{err_3} + k_d \dot{q}_{err_3}) \quad (7)$$

$$\delta_r = -2(k_p q_{err_4} + k_d \dot{q}_{err_4}) \quad (8)$$

If not all axes respond at the same speed, this results in undesired intermediate thrust vectors which disturb the position control in the position control loop.

2.3 Tilt-Twist

In hover, the two rotation angles that influence the position control are the pitch and roll. This is referred to as the tilt angle. The remaining angle is then called twist. For the Nederdrone, the tilt angles are much faster in response time than the twist since the large wings dampen the turn rate around the z-axis a lot and the torque difference of the hover motors is limited. In the presence of turbulence, the twist can even get saturated.

To address this, the tilt should be treated separately from the twist such that they can have differences in response speed. This is described as tilt-twist control [10].

2.3.1 Tilt error

The first part of the tilt-twist method consists in calculating the tilt error. The error is calculated using the rotation matrices to align the current frame with the desired frame

$$\boldsymbol{R}(\boldsymbol{q}) = \begin{bmatrix} q_i^2+q_x^2-q_y^2-q_z^2 & 2(q_xq_y+q_zq_i) & 2(q_xq_z-q_yq_i) \\ 2(q_xq_y-q_zq_i) & q_i^2-q_x^2+q_y^2-q_z^2 & 2(q_yq_z+q_xq_i) \\ 2(q_xq_z+q_yq_i) & 2(q_yq_z-q_xq_i) & q_i^2-q_x^2-q_y^2+q_z^2 \end{bmatrix} \quad (9)$$

Rotation matrices for both the current ($\boldsymbol{R}_c$) and the desired ($\boldsymbol{R}_d$) attitudes are computed as

$$\boldsymbol{R}_d = \boldsymbol{R}(\boldsymbol{q}_d) \quad \boldsymbol{R}_c = \boldsymbol{R}(\boldsymbol{q}_c) \quad (10)$$

The total tilt error can be defined as the shortest rotation between the actual and desired z-axis as illustrated in Figure 3. Note that the axis definition [10] differs from the one used in this work.

$$\boldsymbol{R}_d = \boldsymbol{R}_{err}\boldsymbol{R}_c \quad (11)$$

Equation 11 can be rewritten as

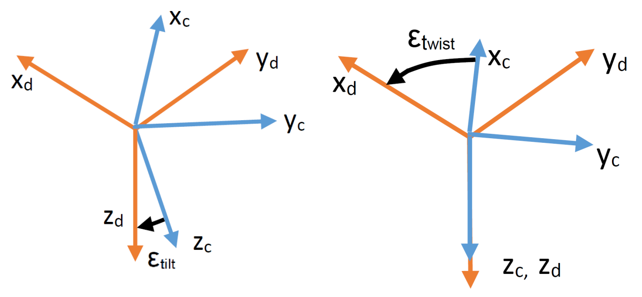


Figure 3: The tilt and twist error definitions. In the tilt plot, the pitch error is 10°, the roll error 5° and the yaw error is 60°. In the twist plot, the pitch error is 0° and the roll error 0°.

$$\boldsymbol{R}_{err} = \boldsymbol{R}_d\boldsymbol{R}_c^\top = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{bmatrix} \tag{12}$$

The third row component of matrix $\boldsymbol{R}_{err}$ then provides the tilt error. The x-component of the tilt error is given by

$$Tilt\ error_1 \triangleq \varepsilon_x = -atan2(r_{3,2}, r_{3,3}) \tag{13}$$

where $atan2$ is the inverse tangent. The y-component is given by

$$Tilt\ error_2 \triangleq \varepsilon_y = atan2(r_{3,1}, r_{3,3}) \tag{14}$$

2.3.2 Twist error

The twist error is calculated as the angle error around the body fixed z-axis. An intermediate coordinate frame is defined that reflects the current attitude after removing the tilt error. This is achieved by using the rotation matrices

$$\boldsymbol{R}_d = \begin{bmatrix} \boldsymbol{r}_{d1} \\ \boldsymbol{r}_{d2} \\ \boldsymbol{r}_{d3} \end{bmatrix} \quad \boldsymbol{R}_c = \begin{bmatrix} \boldsymbol{r}_{c1} \\ \boldsymbol{r}_{c2} \\ \boldsymbol{r}_{c3} \end{bmatrix} \tag{15}$$

where the elements $\boldsymbol{r}$ represent vectors with the body axes expressed in the vehicle frame. The total tilt error then becomes

$$Tilt\ error \triangleq \varepsilon_{tilt} = \cos^{-1}\left(\boldsymbol{r}_{d3}^\top \cdot \boldsymbol{r}_{c3}^\top\right) \tag{16}$$

Next, the unit length axis is defined as

$$k = \frac{\boldsymbol{r}_{c3}^\top \times \boldsymbol{r}_{d3}^\top}{|\boldsymbol{r}_{c3}^\top \times \boldsymbol{r}_{d3}^\top|} \tag{17}$$

As the rotation needs to happen in the vehicle frame, the unit vector k is rotated to the vehicle frame

$$\boldsymbol{v}^b = \boldsymbol{R}_c k = \begin{bmatrix} v_x^b \\ v_y^b \\ v_z^b \end{bmatrix} \tag{18}$$

Then, a rotation matrix is defined which rotates a vector around a vector with a given angle. This is accomplished by using the Rodrigues rotation formula[1], where the angle is ε_{tilt} and the vector is $\boldsymbol{v}^b$. The rotation becomes

$$\boldsymbol{R}_v = \begin{cases} \boldsymbol{I}, & \varepsilon_{tilt} = 0 \\ \boldsymbol{I} - \boldsymbol{v}\sin(\varepsilon_{tilt}) + \boldsymbol{v}^2[1 - \cos(\varepsilon_{tilt})], & \varepsilon_{tilt} \neq 0 \end{cases} \tag{19}$$

where

$$\boldsymbol{v} = \begin{bmatrix} 0 & -v_z^b & v_y^b \\ v_z^b & 0 & -v_x^b \\ -v_y^b & v_x^b & 0 \end{bmatrix} \tag{20}$$

The error is then expressed in the body frame by multiplying $\boldsymbol{R}_v$ with the rotation matrix of the current attitude of the UAV $\boldsymbol{R}_c$

$$\boldsymbol{R}_p = \boldsymbol{R}_v^\top \boldsymbol{R}_c \tag{21}$$

where

$$\boldsymbol{R}_p = \begin{bmatrix} \boldsymbol{r}_{p1} \\ \boldsymbol{r}_{p2} \\ \boldsymbol{r}_{p3} \end{bmatrix} \tag{22}$$

The absolute twist error, illustrated in Figure 3, can be found using the x-components of $\boldsymbol{R}_p$ and $\boldsymbol{R}_d$ with

$$\varepsilon_{twist} = \cos^{-1}(\boldsymbol{r}_{p1}^\top \cdot \boldsymbol{r}_{d1}^\top) \tag{23}$$

To determine the sign of the twist error, the y component of $\boldsymbol{R}_p$ is used in

$$\varepsilon_{sign} = \cos^{-1}(\boldsymbol{r}_{p2}^\top \cdot \boldsymbol{r}_{d1}^\top) \tag{24}$$

Finally, when ε_{twist} is above 90°, the sign of the twist error becomes inverted and is corrected with

$$Twist\ error \triangleq \varepsilon_z = \begin{cases} \varepsilon_{twist}, & \varepsilon_{sign} \leq \frac{\pi}{2} \\ -\varepsilon_{twist}, & \varepsilon_{sign} > \frac{\pi}{2} \end{cases} \tag{25}$$

Together, the total feedback error is set as

$$Feedback\ error \triangleq \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \end{bmatrix} \tag{26}$$

The PD reference generator in the total INDI controller [11] is then written as

$$\delta_a = k_p\varepsilon_x - k_d\dot{\varepsilon}_x \tag{27}$$

$$\delta_e = k_p\varepsilon_y - k_d\dot{\varepsilon}_y \tag{28}$$

$$\delta_r = k_p\varepsilon_z - k_d\dot{\varepsilon}_z \tag{29}$$

By using gains that result in slower reaction on the twist, time-separation of tilt and twist is achieved.

[1]https://mathworld.wolfram.com/RodriguesRotationFormula.html, Oct 2020

2.4 Comparison of quaternion and tilt-twist

Figure 4 illustrates the difference between quaternion feedback tilt-twist feedback on a purely kinematic model with rate-limited yaw control. A simulation is performed where there is a small pitch and roll error and a large yaw error. The rate-limited yaw takes more time to converge than pitch and roll, but more importantly, in quaternion control, the pitch and roll only reach their desired values when the yaw has converged. Since errors in tilt also affect the position control in the outer loop, this will result in larger position errors for the simple quaternion control. The tilt-twist method addresses this problem.

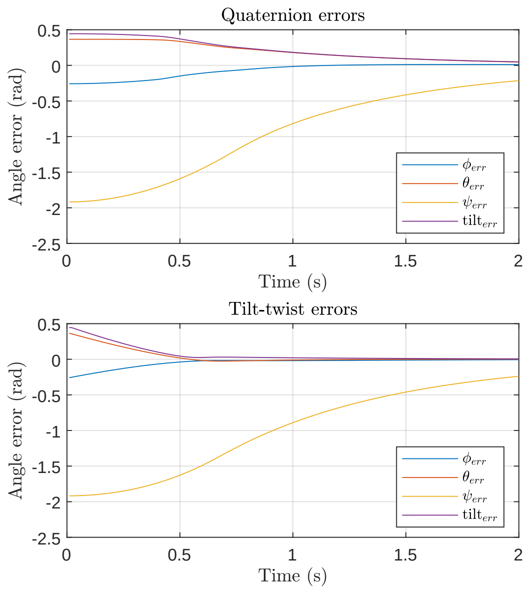


Figure 4: Error handling comparison between tilt-twist and quaternion feedback, in Euler angles. The tilt error is also shown, which is linked to the thrust vector.

2.5 Dynamic Tilt-Twist

In the tilt-twist method, the twist component is always measured around the body fixed z-axis. This is ideal in quadrotors but in hybrid aircraft such as the Nederdrone, when there is wind, the UAV hovers at pitch angles of up to 50 to 70 degrees nose down from hover. In that case, the body tilt axis does not correspond to the lift vector anymore. The goal of tilt-twist—to have the actual lift vector make the shortest rotation—thereby becomes invalid. Moreover, in these conditions, the additional airflow over the main wing, fortunately, improves the achievable turn rate around the x-axis.

To address these conditions, a dynamic tilt-twist controller is introduced. The tilt axis is redefined to be parallel with the gravitational vector. To align the twist vector with the gravitational vector the rotational matrix $\boldsymbol{R}_a$ is used which includes the pitch (θ) and roll (ϕ) angle:

$$\boldsymbol{R}_a(\theta,\phi) = \begin{bmatrix} \cos-\theta & \sin-\theta\sin-\phi & \sin-\theta\cos-\phi \\ 0 & \cos-\phi & -\sin-\phi \\ -\sin-\theta & \cos-\theta\sin-\phi & \cos-\theta\cos-\phi \end{bmatrix} \tag{30}$$

Thereby, the rotation matrices from Equation 10 are redefined as

$$\boldsymbol{R}_d = \boldsymbol{R}_a\boldsymbol{R}(\boldsymbol{q}_d) \tag{31}$$

$$\boldsymbol{R}_c = \boldsymbol{R}_a\boldsymbol{R}(\boldsymbol{q}_c) \tag{32}$$

Then the same tilt-twist controller is used as in the previous section, except for the last step where the actuator deflections need to be compensated again for the dynamic tilt angle using the inverse rotation $\boldsymbol{R}_a^{-1}$

$$Feedback\ error \triangleq \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \end{bmatrix} \boldsymbol{R}_a^{-1} \tag{33}$$

3 SIMULATION AND FLIGHT TEST

The paparazzi autopilot system [13] and simulator with a Nederdrone model are used to perform the simulations and flight tests. The controller is an INDI controller [14] with a modified linear control input. Three different reference generators are compared: quaternion feedback, tilt-twist, and dynamic tilt-twist. The setup is identical for the simulations and the flight test. The tests consisted of flying the Nederdrone back and forth between two waypoints where it needed to hover for three seconds. To induce errors in yaw and highlight the differences in the different controllers, a heading offset ψ_{change} is artificially added each time the Nederdrone leaves a waypoint. Finally, the trajectory errors are compared since optimizing trajectory tracking is the overarching goal.

3.1 Results

3.1.1 Simulation

Before the flight tests, the different types of feedback errors were tested in a simulation. The same controller settings were used for the flight test. The heading offset (ψ_{change}) during the simulation was set to 160°. The simulation results are presented in Figure 5. During the simulation, no wind was taken into account. It can be seen that the quaternion feedback resulted in irregular behavior. The tilt-twist and dynamic tilt-twist methods were much more consistent.

3.1.2 Flight test

Flight tests in windy conditions were performed with the Nederdrone on a path (orange) at about 50° angle with the wind. The heading offset, ψ_{change}, was set to jumps of 45° for the tilt-twist controllers but due to stability issues only to 30° for the quaternion controller (See Figure 6). Higher heading offsets could result in an unsafe flight when the quaternion feedback controller is used.

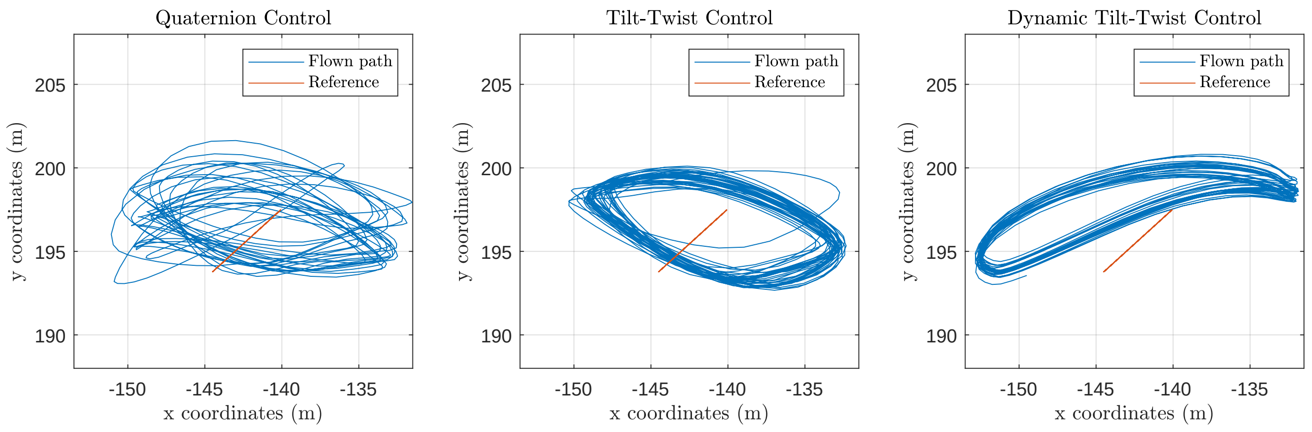


Figure 5: Flight paths during simulation (quaternion, tilt-twist, dynamic tilt-twist). The quaternion feedback method had problems with handling the yaw angle error. Both the tilt-twist and the dynamic tilt-twist method showed more stable behavior.

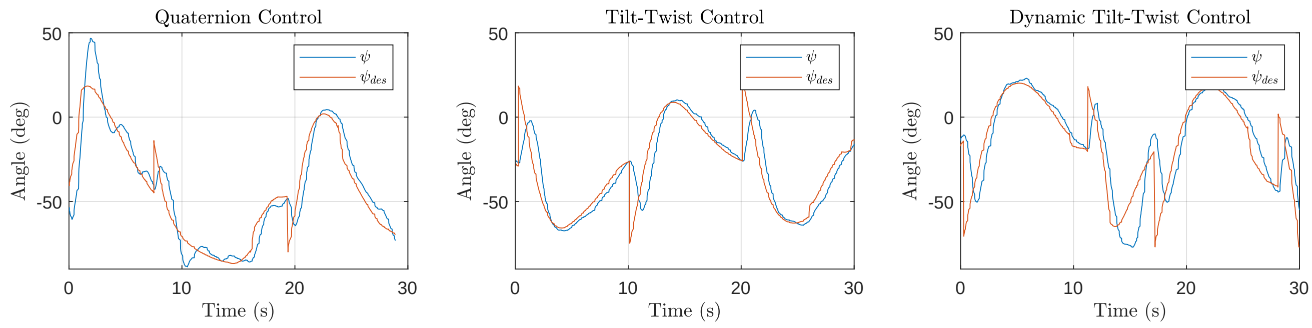


Figure 6: Commanded and actual heading angle ψ where the artificially added jump in commanded heading simulates situations where a large heading error is present.

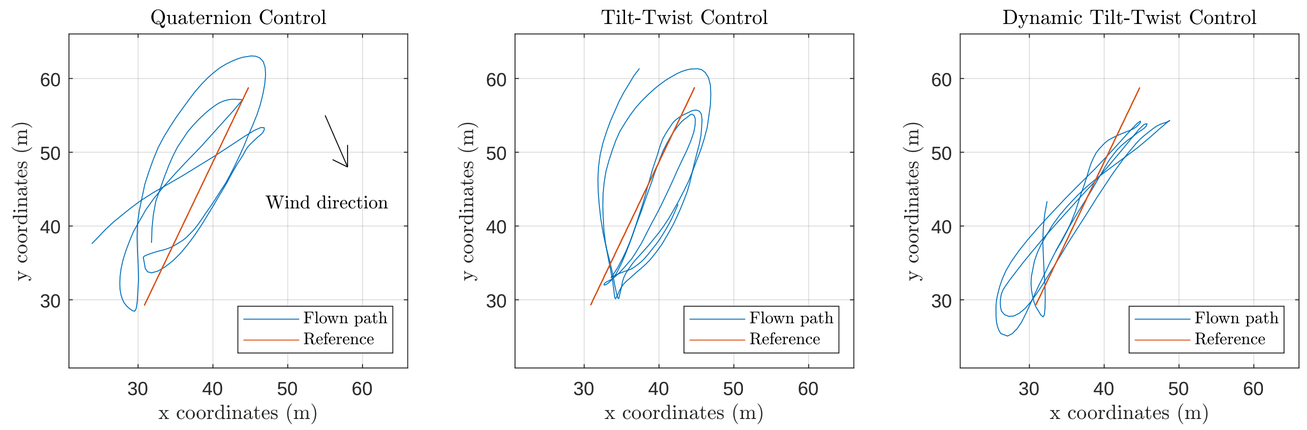


Figure 7: Flown path during flight test, the wind conditions were equal for all flights

The flight paths of the Nederdrone can be seen when using the three controllers in Figure 7. Table 1 shows the numeric comparison of the achieved position accuracy as measured during the same flight in the same conditions by switching the controller in flight. The quaternion controller shows some irregular behavior that depends on the difficulty to reach the desired yaw angle. The tilt-twist method is shown to yield more consistent results than the quaternion controller in the same conditions. Finally, the dynamic tilt-twist method improves the behavior even further.

Table 1: Test results, distance from the reference line.

	Average distance (m)
Quaternion	3.27
Tilt-twist	2.74
Dynamic tilt-twist	2.08

4 Conclusion

This paper presented the combination of INDI control with tilt-twist control and proposed an improvement for hybrid aircraft where the lift vector does not always correspond to the thrust vector, namely the dynamic tilt-twist method. The simulation and the flight test demonstrated that the quaternion feedback method had problems following the required path, whereas the tilt-twist method showed some improvements and the dynamic tilt-twist showed the best results. The dynamic tilt-twist method is suitable for tail-sitters that vary their pitch and roll angles during hover and experience yaw/position control problems.

Acknowledgments

The authors would like to thank the Nederdrone team and sponsors for their support.

References

[1] L. Canetta, G. Mattei, and A. Guanziroli. Exploring commercial UAV market evolution from customer requirements elicitation to collaborative supply network management. In *2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, pages 1016–1022, June 2017.

[2] S. Bouabdallah, P. Murrieri, and R. Siegwart. Design and control of an indoor micro quadrotor. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 5, pages 4393–4398 Vol.5, 2004.

[3] A. Saha, A. Kumar, and A. K. Sahu. FPV drone with GPS used for surveillance in remote areas. In *2017 Third International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pages 62–67, 2017.

[4] K. Feng, W. Li, S. Ge, and F. Pan. Packages delivery based on marker detection for UAVs. In *2020 Chinese Control And Decision Conference (CCDC)*, pages 2094–2099, Aug 2020.

[5] G. Flores and R. Lozano. Transition flight control of the quad-tilting rotor convertible MAV. In *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 789–794, 2013.

[6] K. Muraoka, N. Okada, and D. Kubo. Quad Tilt Wing VTOL UAV: Aerodynamic Characteristics and Prototype Flight. In *AIAA Infotech@Aerospace Conference*, 2012.

[7] C. De Wagter, B. Remes, R. Ruijsink, F. van Tienen, and E. van der Horst. Design and Testing of a Vertical Take-Off and Landing UAV Optimized for Carrying a Hydrogen Fuel Cell with a Pressure Tank. *Unmanned Systems*, 08(04):279–285, 2020.

[8] A. Wang and T. Chan. Estimation of Drag for a Quad-Plane Hybrid Unmanned Aerial Vehicle. In *2017 AIAA Student Conference Region VII-AUAt: Melbourne, Australia*, 11 2017.

[9] C. De Wagter, R. Ruijsink, E. Smeur, K. van Hecke, F. van Tienen, E. van der Horst, and B. Remes. Design, control, and visual navigation of the DelftaCopter VTOL tail-sitter UAV. *Journal of Field Robotics*, pages 937–960, 2018.

[10] T. Matsumoto, K. Kita, R. Suzuki, A. Oosedo, K. Go, A. Konno Y. Hoshino, and M. Uchiyam. A Hovering Control Strategy for a Tail-Sitter VTOL UAV that Increases Stability Against Large Disturbance. In *2010 IEEE International Conference on Robotics and Automation*, Anchorage, AK, USA, May 2010. IEEE.

[11] S. Sieberling, Q. P. Chu, and J. A. Mulder. Robust Flight Control Using Incremental Nonlinear Dynamic Inversion and Angular Acceleration Prediction. *Journal of Guidance, Control, and Dynamics*, 33(6):1732–1742, 2010.

[12] J. Cariño, H. Abaunza, and P. Castillo. Quadrotor quaternion control. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 825–831, June 2015.

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

[14] E. Smeur. *Incremental Control of Hybrid Micro Air Vehicles*. PhD thesis, Delft University of Technology, 2018.

Sliding-mode based Thrust Vector Control for Aircrafts

I. Martinez-Perez[*], R. Garcia-Rodriguez, M.Vega-Navarrete, and L. Ramos-Velasco
Aeronautical Engineering Program and Postgraduate Program in Aerospacial Engineering
Universidad Politécnica Metropolitana de Hidalgo, Mexico

Abstract

In recent years, aircraft research has focused its attention on propulsion systems that control vehicle attitude and flight path by Thrust Vector Control, TVC. The propulsion system with an integrated TVC mechanism is characterized to be provided better maneuverability to the aircraft through moments that allow rotating the flying vehicle helping to the attitude control. This paper proposes a sliding-mode-based TVC for an aircraft, focusing our attention on controlling the angle of attack through the convergence flight-path angle and the pitch angle. Using a rocket as a study system, we firstly present its dynamic model, assuming that the shape of the Earth is an ellipsoid. Then, a sliding-mode-based TVC is proposed to guarantee the aircraft attitude control, regulating the effective angle of attack. Finally, some result simulations show the performance of the proposed controller under different conditions.

1 Introduction

The initiative to make space flights more accessible and cheap has currently experienced rapid development on many topics as aerodynamics, fluids dynamics, propulsion, control, structural dynamics, to name but a few. In recent years, the propulsion systems that include TVC has attracted researchers' attention due to allowing change the flight path, correct a deviation from the desired trajectory, or change the altitude during the powered flight, [1], [2], [3]. In this way, TVC is used to pitch and yaw aircraft controls based on the main rocket nozzle. As part of the thrust system, the gimbal mechanism is in charge to move the nozzle in two or three degrees of freedom through actuators. Thus, for small aircraft, TVC based on electromechanical actuators is the most popular. Structural analysis, gimbal mechanism, control, and sizing are active areas of research of TVC.

In this paper, a thrust vector controller actuating in a single-engine rocket is proposed. Although operations of the rocket flight have many stages, we focus our attention on rocket landing, in particular, the flight control to compensate disturbance forces due to the influence of the environment and the parametric uncertainties of the rocket. Thus, the attitude

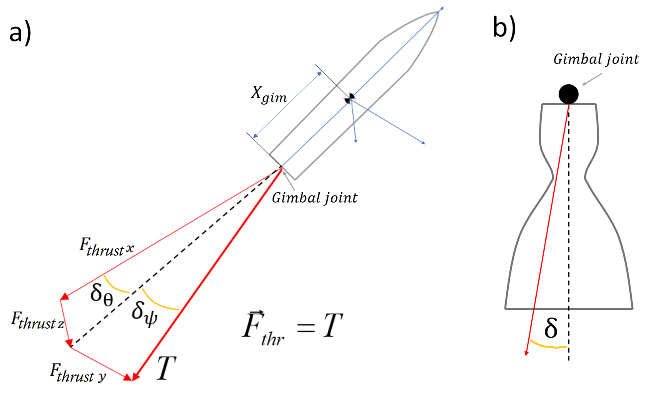


Figure 1: a) Aircraft with an integrated TVC mechanism, b) Gimbal mechanism: A gimbal is essentially a universal joint.

rocket control with TVC is proposed using embedded motion equations of pitch attitude and inertial Z-axis drift position. Furthermore, from aircraft aerodynamics, it's known that the aircraft does not have a straight path to its destination; on the contrary, regularly, there is a slight deviation concerning the angle route or trajectory called the track, while its deviation is known as the angle of drift. Thus, the effects related to the wind must be added with the lifting and dragging to keep our vehicle stable. In this way, assuming some aircraft parameters are unknown and the wind speed is uncertain, a sliding-mode-based TVC is proposed to guarantee the pitch angle and drift position convergence. In addition, some stability conditions are guarantee using the Lyapunov theory. Finally, to validate the proposed approach, some simulations under different conditions are presented. The Sliding Mode Control (SMC) is characterized to provide robustness to parametric uncertainty and external perturbations using high-speed switching feedback control, [4], [5]. Due to its ability to deal with disturbance attenuation and robustness, the SMC is used commonly in flight control design or in combination with other approaches as backstepping or adaptive control. [6], [7], [8], [9], [10], [11].

This paper is organized as follows. In Section 2, the mathematical rocket model is presented with an integrated TVC system. In Section 3, a sliding mode control to guarantee the convergence of angle of attack through the convergence flight-path angle and the pitch angle is proposed. In Section 4, simulation results are presented under different conditions. Finally, some conclusions are presented in Section 5.

[*]Email address: 192220109@upmh.edu.mx

2 Rocket Dynamical Model

In this section, we present the rigid body analysis to obtain the rocket non-linear motion equations.

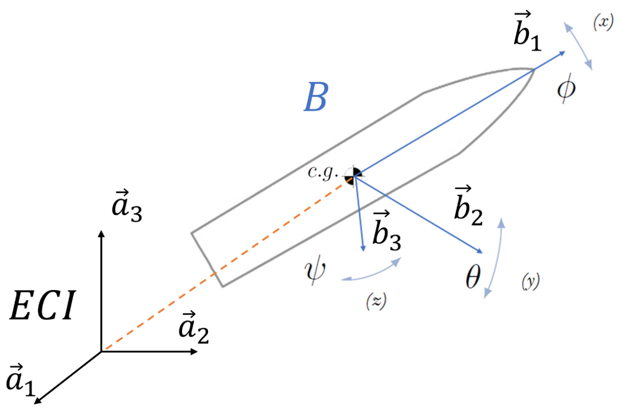


Figure 2: Reference system B (rocket) relative to ECI system.

2.1 *Rotational kinematics*

From the rotational kinematics, we describe the orientation of the aircraft by the Euler angles: roll (ψ), pitch (θ), and yaw (ϕ), see Figure 2, [12]. We start defining two reference systems: the Earth-Centered Inertial system (ECI) and the Body system (B) that refers to the aircraft frame. So, the rotational kinematics of the rocket to ECI frame is defined as:

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \frac{1}{\cos\theta}\begin{bmatrix}\cos\theta & \sin\phi\sin\theta & \cos\phi\sin\theta\\ 0 & \cos\phi\cos\theta & -\sin\phi\cos\theta\\ 0 & \sin\phi & cos\phi\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} \tag{1}$$

where p, q, r represents the components of angular velocity.

2.2 *Translational Dynamics*

In this part, we will define the forces that act on the rocket. Specifically, we refer to the thrust force ($\vec{F}_{thrust}$), aerodynamic force ($\vec{F}_{aer}$) defined concerning the reference system B (Body) while the gravity force ($\vec{F}_g$) respect to ECI reference system. Applying the Second law's Newton (for a constant mass) concerning the ECI reference system, we have that

$$\begin{aligned}\vec{a} &= \begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = \frac{1}{m}\vec{F}\\ &= \frac{1}{m}\left[T^B_{ECI}(\vec{F}_{aer} + \vec{F}_{thrust}) + \vec{F}_g\right]\end{aligned} \tag{2}$$

where T^B_{ECI} is the transformation matrix defined by $T^B_{ECI} = R(x,\phi)R(y,\theta)R(z,\psi)$ and $R(x,\phi), R(y,\theta), R(z,\psi)$ are the rotation matrices corresponding to each axis.

In the following sub-section, we will show how are defined the principal force applied to the rocket, that is, $\vec{F}_{thrust}$, $\vec{F}_{aer}$, and $\vec{F}_g$.

2.2.1 Thrust Force

The thrust delivered by the rocket engine can be calculated by:

$$T = \dot{m}v_e + (P_e - P_0)A_e \tag{3}$$

where $\dot{m}v_e$ is the propellant burned escaping a constant velocity, $(P_e - P_0)A_e$ is the pressure difference, between inside the nozzle (P_e) and outside the nozzle (P_0), on an escape surface (A_e). Assuming that the thrust is constant T, we proceed to determine the thrust vector components, which allows us to define the TVC, see Figure 1a. Using the relationship of

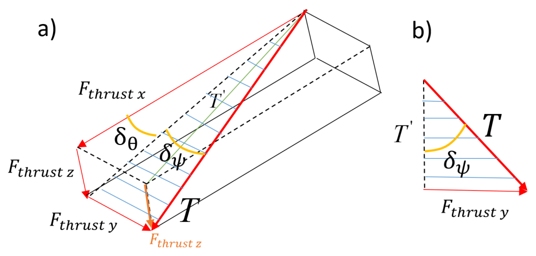


Figure 3: a) $\vec{F}_{thrust}$ Components b) y-component example

right triangles, from Figure 3 we can define the components of the TVC. Thus, the thrust y-component is defined as

$$F_{thrust_y} = T\sin\delta_\psi \tag{4}$$

while the thrust z-component is given as

$$F_{thrust_z} = T\sin\delta_\theta \tag{5}$$

Finally, using that $T' = T\cos\delta_\psi$ we have that thrust x-component is given as

$$\begin{aligned}F_{thrust_x} &= T\prime\cos\delta_\theta\\ F_{thrust_x} &= T\cos\delta_\psi\cos\delta_\theta\end{aligned} \tag{6}$$

In this way, the $\vec{F}_{thrust}$, which include the TVC, is defined as:

$$\vec{F}_{thrust} = \begin{bmatrix}F_{thrust_x}\\ F_{thrust_y}\\ F_{thrust_z}\end{bmatrix} = \begin{bmatrix}T\cos\delta_\psi\cos\delta_\theta\\ T\sin\delta_\psi\\ T\sin\delta_\theta\end{bmatrix} \tag{7}$$

2.2.2 Aerodynamic Forces

When the rocket is in flight, two important aerodynamic forces are generated: the lift $\vec{L}$ produced by the rocket surfaces, and the drag $\vec{D}$ produced by the air resistance of these same surfaces. Assuming a simple geometry of the rocket, we have that the lift and drag forces can be defined as a normal force N, the axial force A, and a lateral slip force S, see Figure 4.

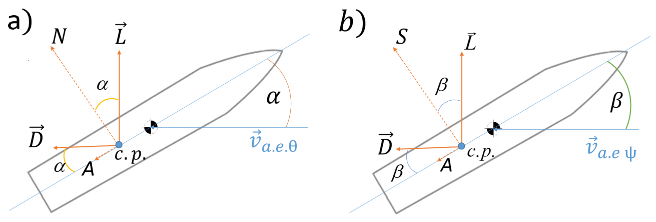


Figure 4: Aerodynamic force: a) xz plane, b) yz plane

Let the axial force A defined directly as $A = \frac{1}{2} C_A \rho V_{air}^2 \cdot surface$ where C_A represents the aerodynamic coefficient given A, ρ the density of the air, V_{air} the airspeed, and $surface$ the total surface where the air affects. For the normal and slip forces, we define them in terms of the real displacement and the wind disturbance (α_w), such that normal force N and the lateral slip force S are defined as:

$$N = N_\alpha \sin\alpha \tag{8}$$

$$S = S_\beta \sin\beta \tag{9}$$

where N_α is the the normal force dependent on the angle of attack α, S_β is the lateral slip force dependent on angle of slip β with $\alpha = \theta + \gamma_\theta + \alpha_{w_\theta}$ and $\beta = \psi + \gamma_\psi + \alpha_{w_\psi}$

In Figure 5 can be seen the actual speed and direction of displacement V, which is defined aerodynamically as the track of the rocket and the angle formed by it and the x axis, or y axis, generally known as the drift angle, $\gamma = \frac{\dot{z}}{V}$. From here, other terms arise, such as the effective wind velocity $\vec{v}_{a.e.}$, the wind disturbance $V_{p.a.}$, for pitch (θ) and roll (ψ) angles, respectively. Thus, the normal force and the lateral slip force affected by the angle of attack are defined as:

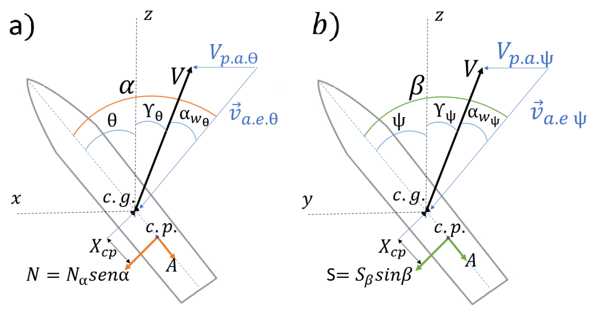


Figure 5: Aerodynamic force with wind disturbance: a) xz plane, b) yz plane

$$N_\alpha = \frac{1}{2} C_N \rho V_{air}^2 \cdot surface \tag{10}$$

$$S_\beta = \frac{1}{2} C_S \rho V_{air}^2 \cdot surface \tag{11}$$

where C_N is the aerodynamic coefficient given the normal force, N, and C_S represents the aerodynamic coefficient given lateral slip force, S. In this way, the aerodynamic forces can be defined as:

$$\vec{F}_{aer} = \begin{bmatrix} F_{aer_x} \\ F_{aer_y} \\ F_{aer_z} \end{bmatrix} = \begin{bmatrix} -A \\ S \\ -N \end{bmatrix} = \begin{bmatrix} -A \\ S_\beta \sin\beta \\ -N_\alpha \sin\alpha \end{bmatrix} \tag{12}$$

REMARK. The components (N,A,S) can be used for ideal aerodynamic performance.

2.2.3 Gravity Force

Unlike other approaches, in this case, we consider the earth's geometry as a WGS84 ellipsoid, proposed in 1984 by the World Geodesic System, currently called Geoid. This assumption is based on the principal rocket mission schedule. In Figure 6 we can observe a cross-section of the ellipsoid earth geometry, where the gravitational acceleration is modeled in geocentric inertial coordinates, [13], getting the gravity as $\mathbf{g} = g_r + g_\lambda$, where each component is defined as:

$$\begin{aligned} g_r &= -\frac{\mu}{r^2}[1 - 3J_2(\frac{R_0}{r})^2 P_2(\cos\Phi_c) \\ &- 4J_3(\frac{R_0}{2})^3 P_3(\cos\Phi_c) - 5J_4(\frac{R_0}{r})^4 P_4(\cos\Phi_c)] \end{aligned} \tag{13}$$

$$\begin{aligned} g_\lambda &= -3\frac{\mu}{r^2}(\frac{R_0}{r})^2 \sin\Phi_c \cos\Phi_c [J_2 + \frac{1}{2}J_3(\frac{R_0}{r}) \sec\Phi_c \\ &(5\cos^2\Phi_c - 1) + \frac{5}{6}J_4(\frac{R_0}{r})^2(7\cos^2\Phi_c - 1)] \end{aligned} \tag{14}$$

where $J_2 = 1.0826e^{-3}$, $J_3 - 2.54e^{-6}$ and $J_4 = -1.61e^{-6}$ are the oblateness terms (dimensionless) or spherical harmonics of an ellipsoid earth, approximated by empirical data to better approximate the earth geometry.

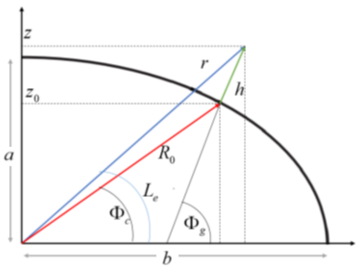


Figure 6: Gravity (Ellipsoidal Earth)

Taking the corresponding transformation of gravity in the geocentric coordinate system to the Cartesian coordinate system about the ECI reference system, [14], we have:

$$g_x = -\frac{\mu}{r^2}\left[1 + \frac{3}{2}j_2(\frac{R_0}{r})^2(1 - 5(\frac{z}{r})^2)\right]\frac{x}{r} \tag{15}$$

$$g_y = -\frac{\mu}{r^2}\left[1 + \frac{3}{2}j_2(\frac{R_0}{r})^2(1 - 5(\frac{z}{r})^2)\right]\frac{y}{r} \tag{16}$$

$$g_z = -\frac{\mu}{r^2}\left[1 + \frac{3}{2}j_2(\frac{R_0}{r})^2(3 - 5(\frac{z}{r})^2)\right]\frac{z}{r} \tag{17}$$

where μ is the universal gravitational parameter, R_0 the distance between the surface location on Earth and the center of Earth, h the height from Earth's surface to the rocket, r the distance between the rocket and the earth center.

Finally, we know that $\vec{F}_g = m\vec{g}$, so the gravity force is defined as:

$$\vec{F}_g = m\begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} = m\begin{bmatrix} -\frac{\mu}{r^2}\left[1+\frac{3}{2}j_2(\frac{R_0}{r})^2(1-5(\frac{z}{r})^2)\right]\frac{x}{r} \\ -\frac{\mu}{r^2}\left[1+\frac{3}{2}j_2(\frac{R_0}{r})^2(1-5(\frac{z}{r})^2)\right]\frac{y}{r} \\ -\frac{\mu}{r^2}\left[1+\frac{3}{2}j_2(\frac{R_0}{r})^2(3-5(\frac{z}{r})^2)\right]\frac{z}{r} \end{bmatrix} \tag{18}$$

2.3 *Rotational Dynamics*

From Euler's second law for a rotating rigid solid, which states that the rate of change of angular momentum $\dot{\vec{L}}$ about a fixed point (or center of mass of the body), is equal to the sum of the moments $\vec{M}$ that act on that body, we have that $\vec{M} = \dot{\vec{L}}$ where the angular momentum $\vec{L}$ is defined as $\vec{L} = \hat{J}\cdot\vec{\omega}$ with $\vec{\omega}$ the angular velocity and $\vec{J}$ the moment of inertia. Thus, the angular acceleration $\vec{\omega}$ is given as, [15]

$$\dot{\vec{\omega}} = \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \hat{J}^{-1}\left[\vec{M} - \vec{\omega}\times\left(\hat{J}\cdot\vec{\omega}\right)\right] \tag{19}$$

Due to the sum of the moments $\vec{M}$ acting on the rocket are produced by the thrust forces and the aerodynamic forces while the gravity acts uniformly over the entire rocket, it does not create a moment, we have that

$$\vec{M} = \vec{M}_{aer} + \vec{M}_{thrust} \tag{20}$$

where $\vec{M}_{aer} = \vec{r}_{c.p.}\times\vec{F}_{aer}$ represents the aerodynamic forces acting on center of pressure ($c.p.$) as the Figure 5, while $\vec{M}_{thrust} = \vec{r}_{gim}\times\vec{F}_{thrust}$ is the thrust force from the gimbal of the nozzle as the Figure 1 a) with $\vec{r}_{c.p.} = \begin{bmatrix}-X_{c.p.} & 0 & 0\end{bmatrix}$ the distance between the $c.p.$ and the $c.g.$, and $\vec{r}_{gim} = \begin{bmatrix}-X_{gim} & 0 & 0\end{bmatrix}$ the distance between the gimbal joint of the nozzle and the $c.g$. The gyro-axis is the center of gravity ($c.g.$).

It is correct to mention that the moment of inertia, produced by the rocket's opposition to moving, calculated by rocket geometry, is variable at each instant of time. So for simpler terms, it will be constant in the rotational dynamics. Now, if we align the axes of the body with the axes of the ECI reference frame, we can reduce the tensor as follows

$$\hat{J} = \begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix} \tag{21}$$

Substituting (21) and (20) in (19), we have that the rotational dynamics is defined as:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{-qr(J_{zz}-J_{yy})}{J_{xx}} \\ \frac{[-X_{c.p.}N_\alpha \sin\alpha + X_{gim}T\sin\delta\theta - pr(J_{xx}-J_{zz})]}{J_{yy}} \\ \frac{[-X_{c.p.}S_\beta \sin\beta - X_{gim}T\sin\delta\psi - pq(J_{yy}-J_{xx})]}{J_{zz}} \end{bmatrix} \tag{22}$$

Finally, we have the complete rocket dynamical model composed of rotational kinematics, translational dynamics, and rotational dynamics. That is, the equations (1), (2) and (22), respectively.

2.4 *Rocket Model Reduction*

In this work, we focus on compensating the wind disturbance by controlling the attack angle by the convergence flight-path angle and pitch angle. Such that the longitudinal rocket dynamics will be used.

Given that depth in y-axis does not exist, since it is a plane, we have that angles ϕ and ψ, and their derivatives, are not considered. So we have that the slip S in this direction is zero. In this way, the control angle of the nozzle δ_ψ which moving in combination with planes xy and yz is zero. Thus, the rocket dynamics on the xz plane is given as:

$$\ddot{x} = [\cos\theta(T\cos\delta\theta - A) - \sin\theta(T\sin\delta\theta - N_\alpha\sin\alpha)]\frac{1}{m} + g_x$$
$$\ddot{z} = [(\sin\theta)(T\cos\delta\theta - A) + (\cos\theta)(T\sin\alpha)]\frac{1}{m} + g_z$$
$$\ddot{\theta} = [-X_{cp}N_\alpha\sin(\alpha)) + X_{gim}T\sin\delta_\theta]/J_{yy}$$

where $\alpha = \theta + \gamma_\theta + \alpha_{w_\theta}$ is the effective angle of attack with $\gamma_\theta = \dot{z}/V$ the flight-path (drift) angle. Due to the study is centered on the attitude control of the rocket during the descent phase, in this paper, we use the equations of height (z) and attitude pitch (θ) only. That is:

$$\begin{aligned} \ddot{z} &= (T\cos(\delta\theta) - A)\frac{\sin(\theta)}{m} \\ &\quad + (T\sin\delta\theta - N_\alpha\sin(\alpha))\frac{\cos(\theta)}{m} \\ &\quad - \frac{\mu}{r^2}[1+\frac{3}{2}j_2(\frac{R_0}{r})^2(3-5(\frac{z}{r})^2)]\frac{z}{r} \end{aligned} \tag{23}$$

$$\ddot{\theta} = [-X_{cp}N_\alpha\sin(\alpha) + X_{gim}T\sin(\delta_\theta)]/J_{yy} \tag{24}$$

Given that TVC is applied to small angles ($\approx 4^\circ - 15^\circ$), we assume $\sin(x) = x$ and $\cos(x) = 1$, we have that rewriting (23)-(24) in state-space, with $\alpha_{w_\theta} = 0$, the rocket dynamics will be defined as

$$\dot{x} = f(*)x + g(*)\delta\theta \tag{25}$$

where $\mathbf{x} = [x_1 = z, x_2 = \dot{z}, x_3 = \theta, x_4 = \dot{\theta}]^T \in \Re^4$, $\delta\theta$ is the rocket thrust vector control,

$$f(*) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ g_z & \frac{N_\alpha}{mV} & \frac{T-A}{m} - \frac{N_\alpha}{m} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{X_{cp}N_\alpha}{V} & 0 & -\frac{X_{cp}N_\alpha}{J_{yy}} \end{bmatrix}$$

$g(*) = [0, \frac{T}{m}, \frac{X_{gim}T}{J_{yy}}]^T$ with $g_z = -\frac{\mu}{r^2}[1+(\frac{9}{2}j_2(\frac{R_0}{r}))^2]\frac{z}{r}$ for $-\frac{15}{2}j_2(\frac{R_0}{r})^2(\frac{z}{r})^2 \approx 0$. Now, we are in conditions to proceed to design a controller to regulate the angle of attack using (25).

3 Controller Design

Notice that (25) is not in a lower triangular form such that approach as backstepping can not be applied. Assuming that some aircraft parameters are not entirely known and the wind speed is uncertain, this paper proposes a sliding-mode-based TVC.

To control the flight-path angle, γ_θ and the pitch angle θ simultaneously through the TVC, we propose a control law as

$$\delta\theta = 1/g_2(-K_d sign(S_1)) \tag{26}$$

where $S_1 = x_1 + x_4$ represents the sliding surface, $g_2 = \frac{X_{gim}T}{J_{yy}}$, and $K_d > 0$ as feedback gain. From SMC theory, to guarantee switching, the condition $S_1\dot{S}_1 < 0$ should be satisfied, [4].

Proposition: Consider (26) to control a flight-path, z angle and pitch angle θ in a rocket. Then convergence to the origin is assured, in finite time $t_f = \frac{|S_1(t_0)|}{\sigma}$, if K_d is large enough for any practical initial conditions.

Proof: Consider the following Lyapunov function $V = 1/2S_1^2$ whose total derivative is given as

$$\begin{aligned}
\dot{V} &= S_1\dot{S}_1 = S_1(\dot{x}_1 + \dot{x}_4) \\
&= S_1(x_2 - \frac{X_{cp}N_\alpha}{V}x_2 - \frac{X_{cp}N_\alpha}{J_{yy}}x_4 + \frac{X_{gim}T}{J_{yy}}\delta\theta) \\
&= S_1([1 - \frac{X_{cp}N_\alpha}{V}]x_2 - \frac{X_{cp}N_\alpha}{J_{yy}}x_4 + \frac{X_{gim}T}{J_{yy}}\delta\theta) \\
&= S_1([1 - \frac{X_{cp}N_\alpha}{V}]x_2 - \frac{X_{cp}N_\alpha}{J_{yy}}x_4 - K_d sign(S_1)) \\
&\leq -K_d|S_1| + |S_1|(\Lambda_1 + \Lambda_2) = -K_d|S_1| + |S_1|\Lambda \\
&\leq -\sigma|S_1| \quad \forall S_1 \neq 0
\end{aligned} \tag{27}$$

where $\sigma = K_d - \Lambda$,$|[1 - \frac{X_{cp}N_\alpha}{V}]||x_2| \leq \Lambda_1$, $\frac{X_{cp}N_{alpha}}{J_{yy}}||x_4 \leq \Lambda_2$, with $\Lambda_1 > 0, \Lambda_2 > 0$, $|x_2| \leq V_{x2}$ and $|x_4| \leq V_{x4}$. Hence, in order to prove that $S_1 \to 0$ in finite time, we can always choose $K_d > \Lambda$ in such a way that $\sigma > 0$ guarantees the existence of a sliding mode at $S_1 = 0$ at time $t_f = \frac{|S_1(t_0)|}{\sigma}$. Thus, a trivial solution that satisfy $S_1 = 0$ is given as $x_1 = x_4 = 0$, i.e. $z = \dot{\theta} = 0$.

4 Numerical Results

In this section, we present the numerical results to show the performance of the proposed controller (26) under different conditions. The simulations were conducted on Python under Windows 10©. The parameters of the rocket used in the simulations are $N_\alpha = 4.46477N$, $A = 6.09525N$, $m = 570 \times 10^3 kg$, $T = 7.605 \times 10^6 N$, $X_{gim} = 21m$, $X_{c.p.} = 10m$, $J_{yy} = 3.2 \times 10^7 kgm^2$, $J_2 = 1.0826 \times 10^{-3}$, $\mu = 3.986 \times 10^14 \frac{m^3}{s^2}$, $R_0 = 6.371 \times 10^6 m$, $r = 6.372 \times 10^6 m$ and $V = 400\frac{m}{s}$, [16]. The simulation's goal is to guarantee the control of the angle of attack by assuring the convergence of the flight path and the pitch angle. The initial conditions used in the simulations are: $z = \dot{z} = \dot{\theta} = 0$ with $\theta = 0.17453$. Finally, for simulations, was used a feedback gain of $K_d = 10$.

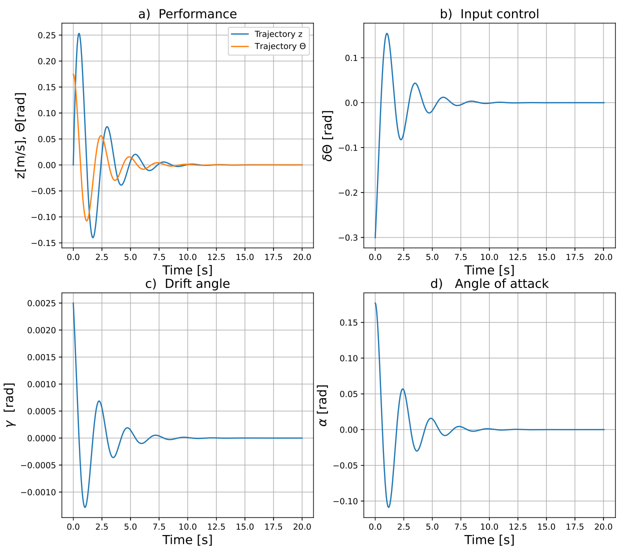


Figure 7: a) Performance under $\alpha_{w\theta} = 0$, b) TVC input control (δ_θ), c) Drift angle (γ), d) Angle of attack (α)

In the first simulation, we consider that the wind disturbance is zero, that is, $\alpha_{w_\theta} = 0$. In Figure 7 a) we observe how the states z, θ converge to the desired value 0, while in Figure 7 b) we show the performance of the TVC input control (δ_θ) within the band of the angle allowed. Figure 7 c) and d) are shown how the drift angle and angle of attack tend to zero, respectively. Notice the convergence relations between z, θ and γ, α, respectively. In the second simulation is proposed two kind of wind disturbances. In the first case the wind disturbance is defined as $\alpha_{w\theta} = \sin(\eta t) + \Delta$ with $\eta = 2$ and $\Delta = 3$, see Figure 8. As in previous case we can notice the robustness of the controller to compensate the parametric uncertainty and the disturbances. In the second case, see Figure 9, we assume that the wind disturbance is defined as a Gaussian noise, that is, $\alpha_{w\theta} = \frac{1}{(2\pi\epsilon^2)}e^{\frac{-(o-\zeta)^2}{(2\epsilon^2)}}$ where $\epsilon = 1$, $\zeta = 0.03$. As previously, we can notice that the controller can compensate the disturbance and guarantee the convergence of the θ and $\dot{z}$ to zero.

A comparative performance indices of the proposed controller and LQR controller is presented in Table 1 where ISE represents the Integral Square Error and IAE the Integral Absolute Error.

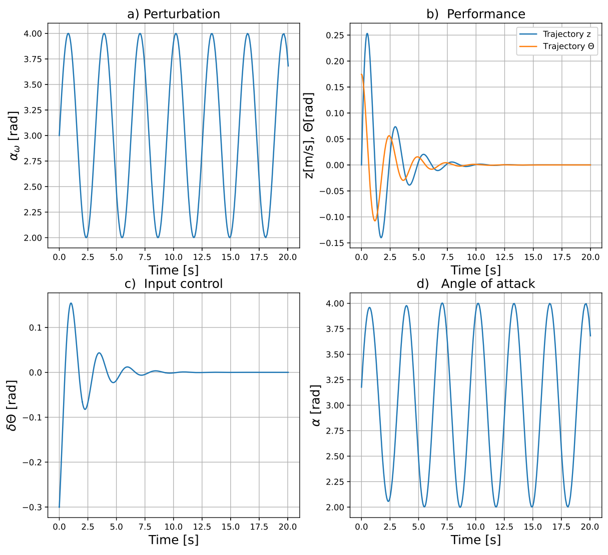


Figure 8: a) Wind disturbance α_{w_θ} as a sinusoidal function, b) Performance including wind disturbance, c) TVC input control, d) Angle of attack

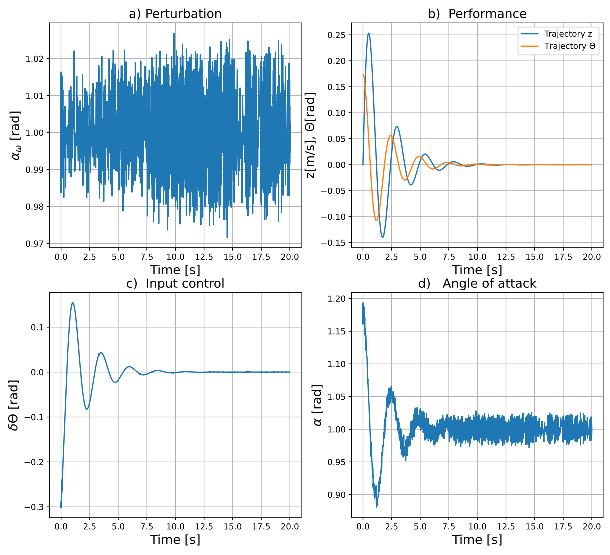


Figure 9: a) Wind disturbance $\alpha_{w\theta}$ as a gaussian function, b) Performance including wind disturbance, c) TVC input control, d) Angle of attack

Table 1. Performance Indices: ISE,IAE

Proposed	$ISE_\theta = 0.0147$	$ISE_z = 0.0756$
	$IAE_\theta = 0.1962$	$IAE_z = 0.4807$
LQR	$ISE_\theta = 0.0439$	$ISE_z = 0.0376$
	$IAE_\theta = 0.3669$	$IAE_z = 0.3242$$

5 CONCLUSION

A complete dynamical model of the rocket, using a TVC, is presented. A reduced dynamical of the rocket is proposed to control the angle of attack, which allows us to reduce the drift in flight through TVC. A model-free sliding-mode-based TVC is presented to guarantee robustness in the presence of parametric uncertainties and wind disturbances. As a part of the Master thesis in Aerospace Engineering at Universidad Politecnica Metropolitana de Hidalgo, Mexico, an experimental rocket is being developed to validate the proposed approaches, see Figure 10.

ACKNOWLEDGEMENTS

This work was partially supported by CONACYT Scholarships with reference number 0055803.

REFERENCES

[1] T. Devlin, R. Dickerhoff, K. Durney, A. Forrest, P. Pansodtee, A. Adabi and M. Teodorescu, ElbowQuad: Thrust Vectoring Quadcopter, 2018 AIAA Information Systems-AIAA Infotech @ Aerospace 8–12 January 2018, Kissimmee, Florida. https://doi.org/10.2514/6.2018-0893

[2] J.P. Silva, C. De Wagter, and G. de Croon, Quadrotor Thrust Vectoring Control with Time and Jerk Optimal Trajectory Planning in Constant Wind Fields, Unmanned Systems, vol. 06(01), pp.15-37, 2018. doi.org/10.1142/S2301385018500024.

[3] X. Wang, B. Zhu, J. Zhu, et al. Thrust vectoring control of vertical/short takeoff and landing aircraft. Sci. China Inf. Sci. 63, 124202 (2020). doi.org/10.1007/s11432-018-9795-y.

[4] J. Y. Hung, W. Gao and J. C. Hung, Variable structure control: a survey, IEEE Transactions on Industrial Electronics, vol. 40(1), pp. 2-22, 1993. doi: 10.1109/41.184817.

[5] R. A. DeCarlo, S. H. Zak and G. P. Matthews, Variable structure control of nonlinear multivariable systems: a tutorial, Proceedings of the IEEE, vol. 76(3), pp. 212-232, 1988. doi: 10.1109/5.4400.

[6] M. Sharma, D.W. Ward, Flight-Path angle control via neuro-adaptive backstepping, AIAA Guidance, Navigation, and Control Conference and Exhibit, 05 August 2002 - 08 August 2002 Monterey, California, USA. doi.org/10.2514/6.2002-4451.

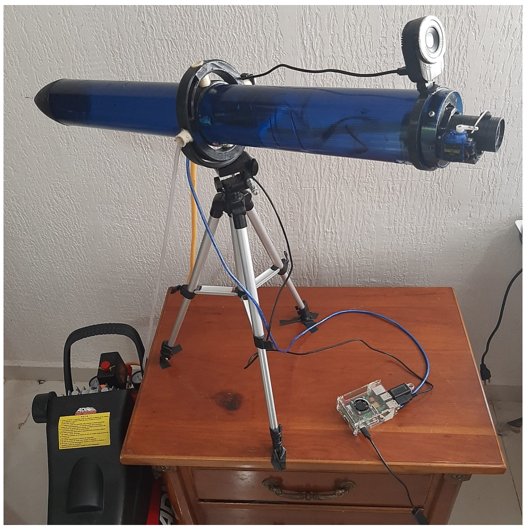


Figure 10: Experimental rocket plane, Universidad Politecnica Metropolitana de Hidalgo.

[7] S.-H. Yoon, Y.-D. Kim, and S.-H. Park, Constrained adaptive backstepping controller design for aircraft landing in wind disturbance and actuator stuck, International Journal of Aeronautical and Space Sciences. The Korean Society for Aeronautical & Space Sciences, 13(1), pp. 74–89. doi: 10.5139/ijass.2012.13.1.74.

[8] B. Choi, HJ. Kim, and Y. Kim, Robust control allocation with adaptive backstepping flight control. Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering, vol. 228(7), pp. 1033-1046, 2014. doi:10.1177/0954410013483687.

[9] Ma. Ran, Q. Wang, D. Hou, C. Dong, Backstepping design of missile guidance and control based on adaptive fuzzy sliding mode control, Chinese Journal of Aeronautics, vol. 27(3), pp. 634-642, 2014. doi.org/10.1016/j.cja.2014.04.007.

[10] Li, Q., Zhang, W., Han, G. and Y. Yang, Adaptive neuro-fuzzy sliding mode control guidance law with impact angle constraint. IET Control Theory Appl., 9: pp. 2115-2123, 2015. doi.org/10.1049/iet-cta.2014.1206

[11] Yuhang Yun, Shengjing Tang, Jie Guo and Yunqiang Zhou, Adaptive sliding-mode guidance with terminal angle constraints based on explicit space engagement model, IEEE Chinese Guidance, Navigation and Control Conference (CGNCC), 2016, pp. 1144-1147, doi: 10.1109/CGNCC.2016.7828949.

[12] W. Bong, D. Wie and W.Mark, Analysis and Design of Launch Vehicle Flight Control Systems, AIAA Guidance, Navigation and Control Conference and Exhibit, 2008. doi:10.2514/6.2008-6291.

[13] Ashish Tewari, Atmospheric and Space Flight Dynamics, 2007 Birkhauser, Boston-Basel-Berlin. ISBN-10: 0-8176-4437-7, doi: 10.1007/978-0-8176-4438-3

[14] Robert F. Stengel, Flight Dynamics, Princeton University Press, 2004, ISBN: 9781680159066, https://doi.org/10.2307/j.ctt1287kgx

[15] Enrique C. del Rio, Cinemática Y Dinámica del Sólido Rigido, Padre Benito Menni-6-2-E 47008 Valladolid, España.

[16] Lusting Matias, Modeling of Launch Vehicle during the Lift-of Phase in Atmosphere, Bachelor's Thesis, Czech Technical University in Prague, May 18, 2017

Quaternion-based attitude sliding mode control with disturbance rejection observer for a quadrotor

J. Ayala-Olivares[*], R. Enriquez-Caldera[†], and J.F. Guerrero-Castellanos[‡]
Intituto Nacional de Astrofísica Óptica y Electrónica, Benemérita Universidad Autónoma de Puebla

ABSTRACT

This work presents a Slide Mode Control (SMC) for the attitude of a quadrotor under unknown disturbances and whose main characteristic is the use of the quaternion mathematics for modeling the system. An extended state observer (ESO) is designed to estimate unknown disturbances and uncertainties. Tests are carried under a previously defined smooth trajectory and the reference quaternion is calculated, and the controller is able to follow the reference to keep the desire orientation. Numerical simulation is shown in order to demonstrate the effectiveness of the proposed control law.

1 INTRODUCTION

1.1 Motivation and Background

A quadrotor is a multi-engine helicopter powered by four engines. These vehicles are easy to build and maintain, and allow very good maneuverability in three-dimensional spaces. In their early days, these vehicles were very poor in terms of computational power, payload capacity and maneuverability. However, given the advances in electronics, which on the hardware side have allowed the miniaturization of its components, and on the software side has allowed the flexibility of the tasks, it has been achieved that today the quadrotors are much smaller, lighter and with such computational power that has resulted in vehicles to perform some tasks autonomously [1].

Two subsystems can be considered when dealing with mathematical models of quadrotors: rotational and translation dynamics [2]. These subsystems provide a cascade structure where translational motion is based on rotational dynamics [3]. Therefore, attitude control is the main part to fulfill trajectory-tracking in the space. This is not a simple task when considering both structural (parametric) and external disturbances. For these reasons, it is necessary to come back to the low-level control problem, i.e., the attitude control problem, and therefore to take into account explicitly in the control design model uncertainties and external disturbances.

In the literature, there are two predominant mathematical models for the quadrotor representation. The most used is obtained by using an Euler's angles representation, allowing a more intuitive understanding of the behavior of the vehicle (see [4, 5]); the second model is based in quaternions offer a compact representation for vehicle's orientation in a 3-D space that is convenient, computationally efficient, and accurate [6, 7, 8].

Both models have disadvantages, for example: the model with Newton-Euler equations is a system of 12 nonlinear differential equations and, in general, to solve them requires a high computational cost and the consequent loss of information in the process due to the numerical methods applied; on the other hand, the model with quaternions is difficult to interpret due to the abstract nature of its theory [9].

Several linear control approaches, such as PID ([10]), Linear Quadratic Regulator (LQR) ([11]) and Linear Quadratic Gaussian (LQG) ([12]), have been proposed in the literature and applied for attitude stabilization and/or altitude tracking of quadrotors. However, these methods can impose limitations on application of quadrotors for extended flight complex flight trajectories where the system is no longer linear. In addition, the performances on tracking trajectories of these control laws are not satisfactory enough [13, 14].

To overcome the linear controllers limitations, there are nonlinear control alternatives, such as Backstepping [15, 16, 17], Feedback linearization [18, 19], Model predictive control [20, 21, 22], among others. These control techniques have shown good performance against sinusoidal wind disturbance but increasing the cost of computational resources [23].

The Sliding Mode Control has been applied extensively to control quadrotors. The primary advantages of SMC are: 1. Fast response and good transient performance. 2. Its robustness against a large class of perturbations or model uncertainties. 3. The possibility of stabilizing some complex nonlinear systems which are difficult to stabilize by continuous state feedback laws [24]. The SMC is a high frequency switching control that causes chattering, an undesired phenomenon which leads towards loss of energy, unmodeled dynamics and even actuator destruction [25].

Furthermore, to enhance performance robustness, an Extended State Observer can be implemented. An ESO can estimate both, unknown states of system and the "total disturbances" that lump the adverse effects on the output using limited model information [26].

[*]Email address: jrayala@inaoe.mx
[†]Email address: rogerio@inaoe.mx
[‡]Email address: fermi.guerrero@correo.buap.mx

1.2 Contributions

The development of a SMC for attitude tracking of a quadrotor with a ESO applied to the quaternion-based model, adding disturbances to the inertial matrix and input control torques.

This paper is organized as follows. The basic concepts of quaternion theory and dynamic model of a quadrotor are presented in Sect.2. In Sect.3, the position control is described, following by the SMC concepts and the attitude controller is designed, later, an ESO is proposed and designed. The simulation results of the proposed controller strategies are presented in Sect.4. Finally, conclusions are included in Sect.5.

2 Quaternion Background and Quadrotor Modeling

A quaternion background is shown next, and the quadrotor model based on quaternions as well.

2.1 Quaternions

A quaternion can be thought as a composite of a scalar $q_0 \in \mathbb{R}$ and an ordinary vector $\mathbf{q}_v = (q_1\ q_2\ q_3)^T \in \mathbb{R}^3$, that is, $\mathbf{q} = (q_0\ \mathbf{q}_v^T)^T$ or as a complex number with three different imaginary parts, *i.e.*, a hypercomplex number [27]. It is represented as follows

$$\mathbf{q} := \begin{pmatrix} q_0 \\ \mathbf{q}_v \end{pmatrix} = q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3 \in \mathbb{H} \tag{1}$$

with $\{1, \mathbf{i}, \mathbf{j}, \mathbf{k}\}$ being a canonical basis of the hypercomplex space $\mathbb{H}$ [7]. The norm of a quaternion is

$$||\mathbf{q}|| = \sqrt{q_0^2 + q_1^2 + q_4^2 + q_3^2} \tag{2}$$

A unit quaternion, with a norm equal to 1, can be used to represent a rotation by an angle β about the axis defined by the unit vector $\mathbf{e} = (e_1\ e_2\ e_3)^T$, that is

$$\mathbf{q} := e^{\frac{1}{2}\beta\mathbf{e}} = \cos\frac{\beta}{2} + \mathbf{e}\sin\frac{\beta}{2} \tag{3}$$

This expression, known as the Euler-Rodrigues formula is the exponential mapping of the axis-angle representation of a rotation. Since the n-dimensional unit sphere embedded in $\mathbb{R}^{n+1}$ is denoted as $\mathbb{S}^n = \{\mathbf{x} \in \mathbb{R}^{n+1} : \mathbf{x}^T\mathbf{x} = 1\}$ then $\mathbf{q} \in \mathbb{S}^3$. Furthermore, $\mathbf{q}$ represents an element of $SO(3)$ through the map $\mathbf{R} : \mathbb{S}^3 \to SO(3)$ defined as:

$$\mathbf{R}(\mathbf{q}) := I_3 + 2q_0[\mathbf{q}_v^\times] + 2[\mathbf{q}_v^\times]^2 \tag{4}$$

$\mathbf{R} \in SO(3) = \{\mathbf{R} \in \mathbb{R}^{3\times3} : \mathbf{R}^T\mathbf{R} = I_3, \det(\mathbf{R}) = 1\}$ is the matrix that rotates the coordinates of a point from frame $\mathbf{E}^b$ to frame $\mathbf{E}^f$ with I_3 as the 3×3 identity matrix. $[\mathbf{r}^\times]$ is the skew-symmetric matrix associated to vector $\mathbf{r}$.
The sum and subtraction of quaternions is performed by separate addition of their four parts. A vector can be converted to quaternion by setting the scalar part to zero and replacing the vector part of the quaternion by the corresponding values of the vector. The quaternion product is defined as

$$\mathbf{q} \otimes \mathbf{r} = (q_0 r_0 - \mathbf{q}_v \cdot \mathbf{r}_v) + (q_0\mathbf{r}_v + r_0\mathbf{q}_v + \mathbf{q}_v \times \mathbf{r}_v) \tag{5}$$

The conjugate of a unit quaternion is defined as $\mathbf{q}^* = q_0 - \mathbf{q}_v$.

$$\mathbf{q}^* = q_0 - \mathbf{q}_v \tag{6}$$

A quaternion can be used as rotation operator for a vector between two different frames. Considering $\mathbf{p}_v$ as a 3D vector in a given reference frame $\mathbf{E}^f$ and $\mathbf{p}'_v$ as the same vector in a new different frame, for instance, $\mathbf{E}^b$. Then, the quaternion $\mathbf{p} = (0\ \mathbf{p}_v^T)^T$ can be transform in $\mathbf{p}'$, and vice-versa through

$$\mathbf{p}' = \mathbf{q}^* \otimes \mathbf{p} \otimes \mathbf{q}, \quad \mathbf{p} = \mathbf{q} \otimes \mathbf{p} \otimes \mathbf{q}^* \tag{7}$$

The properties of the logarithm in the unit quaternion are useful to obtain the equivalent axis-angle notation. For unitary quaternions the logarithmic mapping is given by

$$\ln(\mathbf{q}) = \begin{cases} \frac{\mathbf{q}_v}{\|\mathbf{q}_v\|} \arccos\ q_0, & \|\mathbf{q}_v\| \neq 0 \\ (0\ 0\ 0)^T & \|\mathbf{q}_v\| = 0 \end{cases} \tag{8}$$

With this mapping you can change any unitary quaternion to its axis-angle representation as follows [7]

$$\begin{aligned} \boldsymbol{\beta}_v &= \mathbf{e}\beta = 2\ln(\mathbf{q}) \in \mathbb{R}^3 \\ \dot{\boldsymbol{\beta}}_v &= \boldsymbol{\omega}_v \in \mathbb{R}^3 \end{aligned} \tag{9}$$

where $\boldsymbol{\beta}_v$ is the vector that represents the axis-angle notation, $\mathbf{e} = \frac{\mathbf{q}_v}{\|\mathbf{q}_v\|}$ describes the unit axis about which the rotation is applied, $\|\beta\|$ represents the magnitude of the rotation. $\boldsymbol{\omega}_v$ is the angular velocity vector of the body coordinate frame $\mathbf{E}^b$ relative to the inertial coordinate frame $\mathbf{E}^f$ expressed in $\mathbf{E}^b$.
The attitude error is used to quantify the mismatch between two attitudes. If $\mathbf{q}$ defines the current attitude quaternion and $\mathbf{q}_d$ the desired quaternion, *i.e.*, the desired orientation, then the quaternion that represents the attitude error between the current orientation and the desired one is given by [28]:

$$\tilde{\mathbf{q}} = \mathbf{q}_d^* \otimes \mathbf{q} = \begin{pmatrix} \tilde{q}_0 & \tilde{\mathbf{q}}_v^T \end{pmatrix}^T \tag{10}$$

When the current quaternion $\mathbf{q}$ reaches the desired one $\mathbf{q}_d$, the quaternion error becomes $\tilde{\mathbf{q}} = \begin{pmatrix} \pm 1 & 0^T \end{pmatrix}^T$, *i.e.*, there exist two equilibria which have to be considered in the stability analysis [29].

2.2 Quadrotor Dynamics

A diagram of the quadrotor studied in this paper is shown in Fig. 1, where the inertial frame and the body frame are represented by $\mathbf{E}^f$ and $\mathbf{E}^b$, respectively. Let define the position vector $\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^T$. Then, the related quaternion is given by $\xi = [0\ \mathbf{p}^T]^T$

According to [7, 30, 31], the equations of motion of a quadrotor using quaternions are

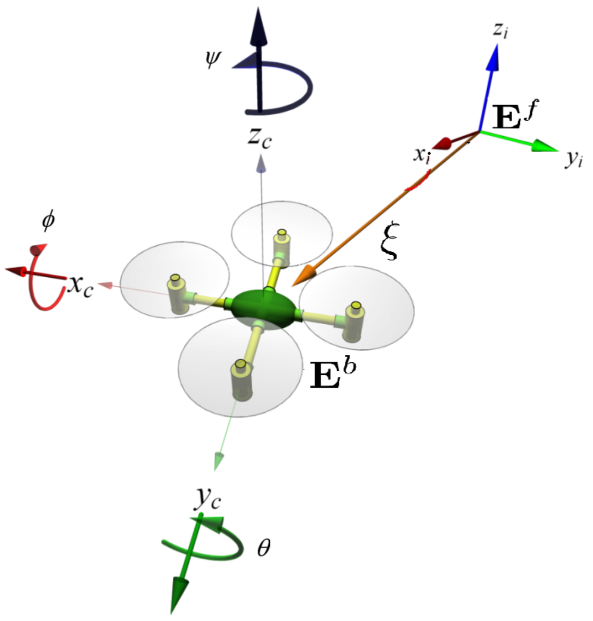


Figure 1: Quadrotor diagram.

$$\dot{X} = \frac{d}{dt}\begin{bmatrix} \xi \\ \dot{\xi} \\ \mathbf{q} \\ \omega \end{bmatrix} = \begin{bmatrix} \dot{\xi} \\ \mathbf{q} \otimes \frac{F_{th}}{m} \otimes \mathbf{q}^* + \bar{g} \\ \frac{1}{2}\mathbf{q} \otimes \boldsymbol{\omega} \\ J^{-1}(\tau - \boldsymbol{\omega}_v \times J\boldsymbol{\omega}_v) \end{bmatrix} \tag{11}$$

where $\mathbf{q}$ represents the vehicle attitude, using a unit quaternion, with respect to the inertial frame, $\boldsymbol{\omega} = [0 \ \boldsymbol{\omega}_v]^T$ and $F_{\text{th}} = \begin{bmatrix} 0 & 0 & 0 & \sum_{i=1}^4 f_i \end{bmatrix}^T$ describes the total thrust force applied to the body in the inertial frame, and the angular velocity. J represents the inertia matrix with respect to the body-fixed frame, and the total torque τ is given by

$$\tau = \begin{bmatrix} l\,(f_1 + f_4 - f_2 - f_3) \\ l\,(f_1 + f_4 - f_2 - f_3) \\ \sum_{i=1}^4 (-1)^{i+1}\tau_i \end{bmatrix}, \tag{12}$$

where l represents the distance between any motor and the center of mass of the vehicle, f_i represents the force generated in the rotor i, with $i \in \{1, 2, 3, 4\}$. $\bar{g} = [0\ 0\ 0\ g]^T$ is the gravity vector.

Applying the logarithmic mapping, the system model is as follows

$$\dot{X} = \frac{d}{d\,t}\begin{bmatrix} \xi \\ \dot{\xi} \\ \boldsymbol{\beta}_v \\ \dot{\boldsymbol{\beta}}_v \end{bmatrix} = \begin{bmatrix} \dot{\xi} \\ \mathbf{q} \otimes \frac{F_{\text{th}}}{m} \otimes \mathbf{q}^* + \bar{g} \\ \dot{\boldsymbol{\beta}}_v \\ J^{-1}\left(\tau - \dot{\boldsymbol{\beta}}_v \times J\dot{\boldsymbol{\beta}}_v\right) \end{bmatrix} \tag{13}$$

3 Quadrotor control

In this section, the control strategy is described, as is shown in the Fig. 2, position control allows the vehicle to follow a defined trajectory, then the attitude control is designed to achieve the necessary orientation given by the position control.

3.1 *Position Control*

The position dynamics subsystem of the quadrotor can be written as

$$\dot{\xi} = \frac{d}{dt}\begin{bmatrix} \xi \\ \dot{\xi} \end{bmatrix} = \begin{bmatrix} \dot{\xi} \\ m^{-1}F^I_{\text{th}} \end{bmatrix} \tag{14}$$

where $F^I_{\text{th}} = \mathbf{q} \otimes F_{\text{th}} \otimes \mathbf{q}^* + m\bar{g}$ is a force that can be designed such that the vehicle reach a desired position. Since Equation 14 is a linear system, the control law F^I_{th} is proposed as

$$F^I_{\text{th}} = \ddot{\xi}_d + k_1\tilde{\xi} + k_2\dot{\tilde{\xi}} \tag{15}$$

where ξ_d is the desired position, $\tilde{\xi} = \xi_d - \xi$ is the position error, with k_1 and k_2 are the control gains. The error dynamics is given by

$$\frac{d}{dt}\begin{bmatrix} \tilde{\xi} \\ \dot{\tilde{\xi}} \end{bmatrix} = \begin{bmatrix} \dot{\tilde{\xi}} \\ \ddot{\xi}_d - (\ddot{\xi}_d - k_1\tilde{\xi} + k_2\dot{\tilde{\xi}}) \end{bmatrix} \tag{16}$$

rewriting equation (16)

$$\frac{d}{dt}\begin{bmatrix} \tilde{\xi} \\ \dot{\tilde{\xi}} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ k_1 & k_2 \end{bmatrix}}_{A}\begin{bmatrix} \tilde{\xi} \\ \dot{\tilde{\xi}} \end{bmatrix} \tag{17}$$

if $k_1, k_2 > 0$ then A is Hurwitz and the trajectories of the error dynamics converge asymptotically to the origin of the vector space error, that is, $\tilde{\xi}, \dot{\tilde{\xi}} \ \to 0$, when $t \to \infty$.

3.2 *Basic concepts of Slide mode control*

The SMC is a type of Variable Structure Control (VSC). Its basic idea is to attract the system states towards a surface, called sliding surface, suitably chosen and design a stabilizing control law that keeps the system states on such a surface. For the choice of the sliding surface shape, the general form of equation (18) was proposed by Stoline and Li in [13]:

$$S(x) = \left(\lambda_x + \frac{d}{dt}\right)^{q-1} e(x) \tag{18}$$

where x denotes the variable control (state), $e(x)$ is the tracking error defined as $e(x) = x - xd$, λ_x is a positive constant that interprets the dynamics of the surface and q is the relative degree of the sliding mode controller.

Attractiveness is the condition under which the state trajectory will reach the sliding surface. There are two types of conditions of access to the sliding surface. In this paper, we will use the Lyapunov based approach. It consists of make

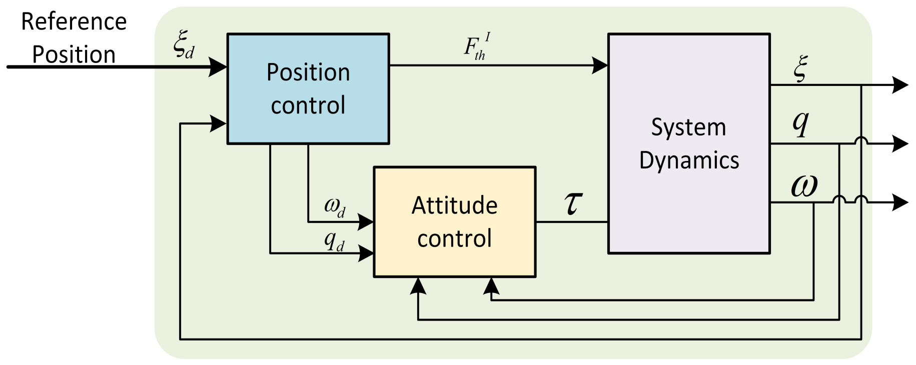


Figure 2: Control Block Diagram.

a positive scalar function, given by equation (19) and called Lyapunov candidate function, for the system state variables and then choose the control law that will decrease this function:

$$\dot{V}(x) < 0, \text{ with } V(x) > 0 \tag{19}$$

In this case, the Lyapunov function can be chosen as:

$$V(x) = \frac{1}{2}S(x)^2 \tag{20}$$

The derivative of this above function is negative when the following expression is checked:

$$S(x)\dot{S_(}x) < 0 \tag{21}$$

The purpose is to force the system state trajectories to reach the sliding surface and stay on it despite the presence of uncertainty. The sliding control law contains two terms as follows:

$$u(t) = u_{eq}(t) + u_D(t) \tag{22}$$

where $u_{eq}(t)$ denotes the equivalent control, which is a way to determine the behavior of the system when an ideal sliding regime is established. it is calculated from the following invariance condition of the surface:

$$\begin{cases} S(x,t) = 0 \\ \dot{S}(x,t) = 0 \end{cases} \tag{23}$$

and $u_D(t)$ is a discontinuous function calculated by checking the condition of the attractiveness. It is useful to compensate the uncertainties of the model and often defined as follows:

$$u_D(t) = -K\text{sign}(S(t)) \tag{24}$$

where K is a positive control parameter and $\text{sign}(\cdot)$ is the sign operator.

3.3 *Attitude Control design*

For the attitude control, we use the rotational motion model given by equation (13), we take the error quaternion $\tilde{\mathbf{q}}$ from Equation 10 and applying the logarithm mapping to $\tilde{\mathbf{q}}$ we get

$$\begin{array}{c} \tilde{\mathbf{q}} = \mathbf{q}_d^* \otimes \mathbf{q} \\ \tilde{\boldsymbol{\beta}}_v = 2\ln(\tilde{\mathbf{q}}) \end{array} \tag{25}$$

$\mathbf{q}_d$ is obtained from the shortest rotation between F_{th} and F_{th}^I vectors as unitary vectors. According to [7] $\mathbf{q}_d$ is defined as

$$\mathbf{q}_d = \exp\left(\ln\left(F_{\text{th}}^I \otimes F_{\text{th}}^*\right)/2\right) \otimes \exp\left(\mathbf{q}_{\psi_d}\right)/2 \tag{26}$$

where $\mathbf{q}_{\psi_d} = \begin{bmatrix} 0 & 0 & 0 & \psi_d \end{bmatrix}^T$ is the desired rotation around the z axis.

The sliding surface is chosen based on the tracking error, such as:

$$S = \dot{\tilde{\boldsymbol{\beta}}}_v + \lambda\tilde{\boldsymbol{\beta}}_v \tag{27}$$

Deriving S, we get

$$\dot{S} = \dot{\omega}_e + \lambda\omega_e \tag{28}$$

where $\omega_e = \omega - \omega_d$ is the rotation velocity error, and ω_d is the desired rotation velocity which is defined as

$$\omega_d = 2\frac{d}{dt}(\ln \mathbf{q}_d) \tag{29}$$

Substituting the model values on equation (28), we get

$$\begin{array}{l} \dot{S} = \dot{\omega} - \dot{\omega}_d + \lambda(w - w_d) \\ \dot{S} = J^{-1}(\tau - \omega \times J\omega) - \dot{\omega}_d + \lambda(w - w_d) \end{array}$$

Finally, the control law is obtained using Equation 22:

$$\begin{array}{c} \tau = J(-(-J^{-1}\omega \times J\omega - \dot{\omega}_d + \lambda(\omega - \omega_d)) + u_D) \\ u_D = -K\text{sign}(S) \end{array}$$

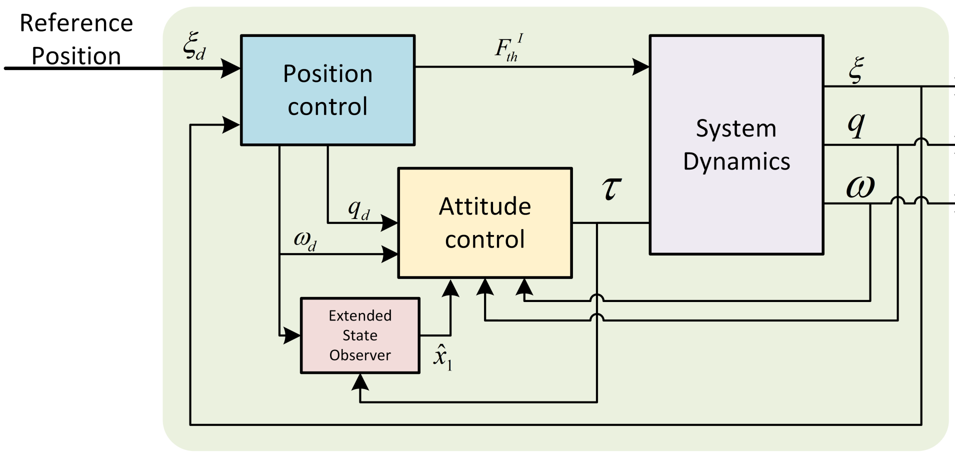


Figure 3: ESO attitude Control Block Diagram.

3.4 *ESO design*

The ESO of quadrotor UAV system should be designed to estimate the model uncertainties and external disturbances [32]. Fig. 3 shows the ESO implementation in the attitude control loop.

The attitude subsystem can be written as

$$\begin{bmatrix} \mathbf{q} \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\mathbf{q} \otimes \omega \\ J^{-1}(\tau - \omega_v \times J\omega_v) \end{bmatrix} \tag{30}$$

In order to use the ESO for the estimation of uncertainties in the inertia matrix and unknown disturbances, ESO is defined as:

$$\begin{aligned} \dot{\hat{x}}_1 &= \hat{x}_2 + l_2(x_1 - \hat{x}_1) - J^{-1}\tau \\ \dot{\hat{x}}_2 &= l_1(x_1 - \hat{x}_1) \end{aligned} \tag{31}$$

where $\hat{x}_1$ is the estimation of the angular velocity ω, and $\hat{x}_2$ is the estimated value of an unknown disturbance d. l_1 and l_2 are the observer tuning parameters.

4 Results

In this section, the proposed control strategy for the quadrotor attitude stabilization is implemented in order to verify his validity and efficiency. For the numerical simulation, the following parameters are using

$$m = 1.3\,\text{kg},\ J = \begin{bmatrix} 0.177 & 0 & 0 \\ 0 & 0.177 & 0 \\ 0 & 0 & 0.354 \end{bmatrix} \text{kg m}^2 \tag{32}$$

For the position controller, the control gains were empirically selected as: $k_1 = -24$ and $k_2 = -12$. In the same manner, the attitude control gains λ and K were selected as

$$\lambda = 10,\ K = 50 \tag{33}$$

4.1 *SMC sans ESO*

The dynamic model was coded using MATLAB, the chosen trajectory was a spiral as is shown in Fig. 4 with the following initial conditions: $\xi_d = [5\,5\,0]^T$m, $\xi = [0\,0\,0]^T$m and $[\phi\,\theta\,\psi]^T = [0\,0\,0]^T$rad. In addition, a uniform noise, $\pm 20\%$ of nominal value, is added to the inertial matrix and a sine wave $d = 0.1\sin(t)$ is applied as a disturbance in the control torques τ_1 and τ_2 signals @t=50s.

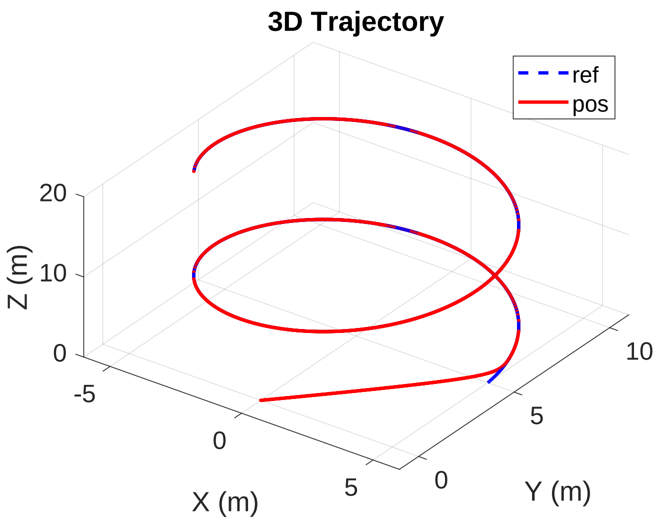


Figure 4: 3D spiral trajectory.

The position errors are shown in the Fig. 5

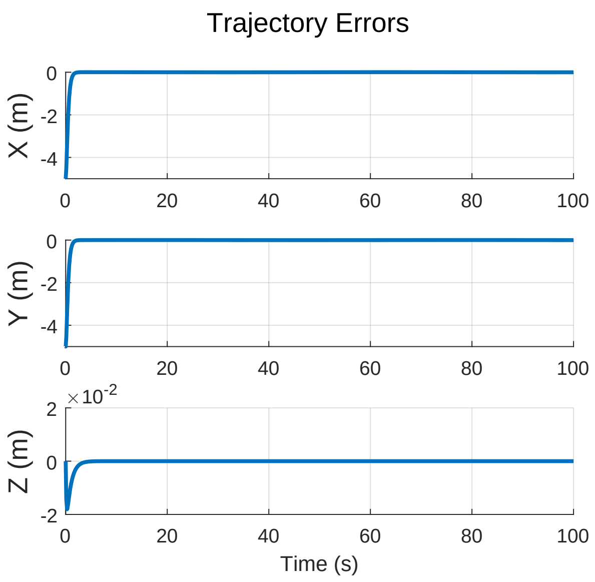


Figure 5: Errors in 3D trajectory.

The desired quaternion and the vehicle's attitude quaternion are shown in the Fig. 6.

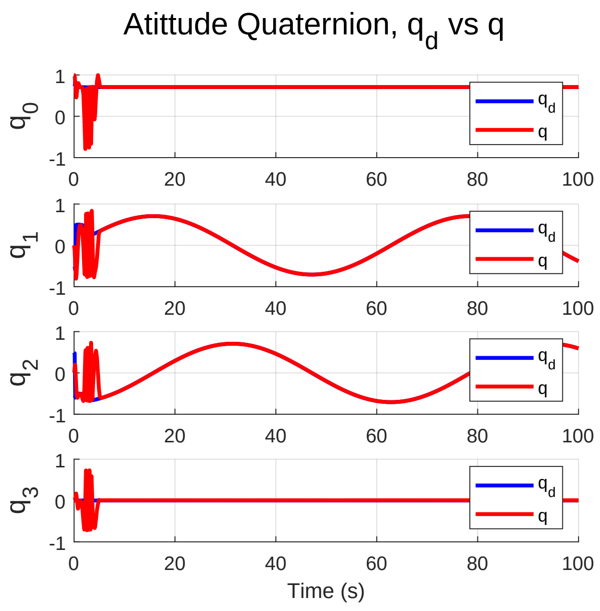


Figure 6: Desired quaternion ($\mathbf{q}_d$) vs Attitude quaternion ($\mathbf{q}$).

The $\tilde{\beta}_v$ angle is shown in Fig. 7, where the SMC is able to compensate the effect of the disturbance and keep the error low.

The torques for the attitude stabilization of the vehicle are shown in the Fig. 8, these were limited to ± 1 N·m

4.2 SMC with ESO

For this case, the same initial conditions were applied, and using the following control gain: $l_1 = 250$, $l_2 = 400$, we get the trajectory shown in Fig. 9

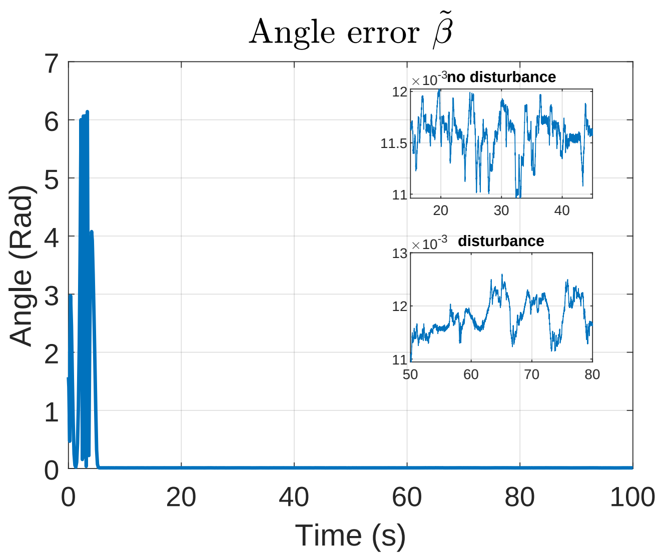


Figure 7: Error angle $\tilde{\beta}$.

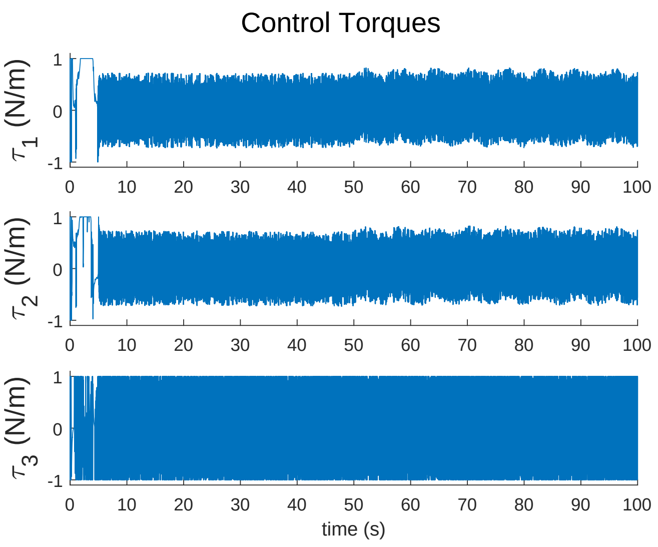


Figure 8: Control torques.

The trajectory errors are shown in Fig. 10.

The desired quaternion and the vehicle's attitude quaternion are shown in the Fig. 11.

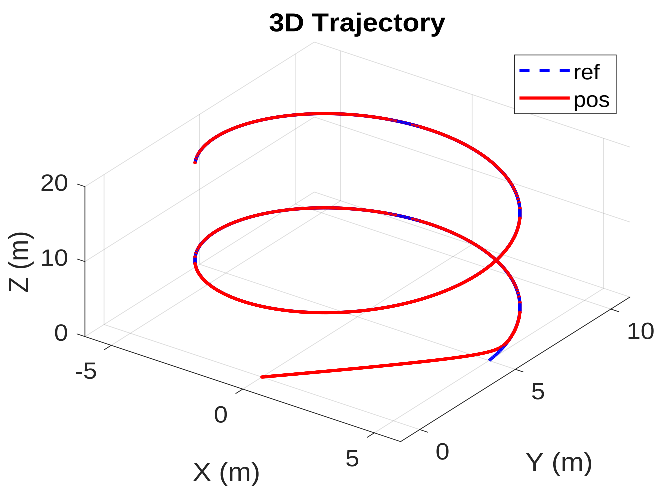


Figure 9: 3D spiral trajectory.

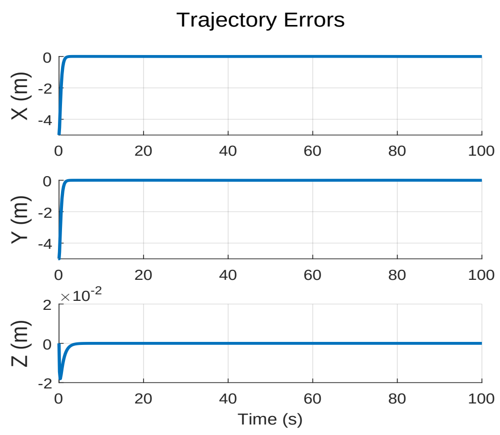


Figure 10: Errors in 3D trajectory.

The $\tilde{\beta}$ angle is shown in Fig. 12, as in the previous case, the error doesn't increase significantly when the disturbance in the input torques appears. The ESO keeps the error an order of magnitude lower than the previous case.

The torques for the attitude stabilization of the vehicle are shown in the Fig. 13, these also were limited to ± 1 N·m

Finally, the Integral Square Error (ISE) is shown in the Fig. 14, where an error of two orders of magnitude smaller error is observed when the ESO is added to the system

5 CONCLUSION

In this paper, we worked with the problem of the attitude stabilization and tracking of a quadrotor vehicle using a nonlinear sliding mode control approach. In addition, an ESO were proposed to estimate the inertia matrix uncertainties and unknown disturbances. Several simulations results are carried out in order to show the effectiveness of the proposed nonlinear control strategies and proving that an ESO is useful when there are uncertainties in the vehicle parameters.

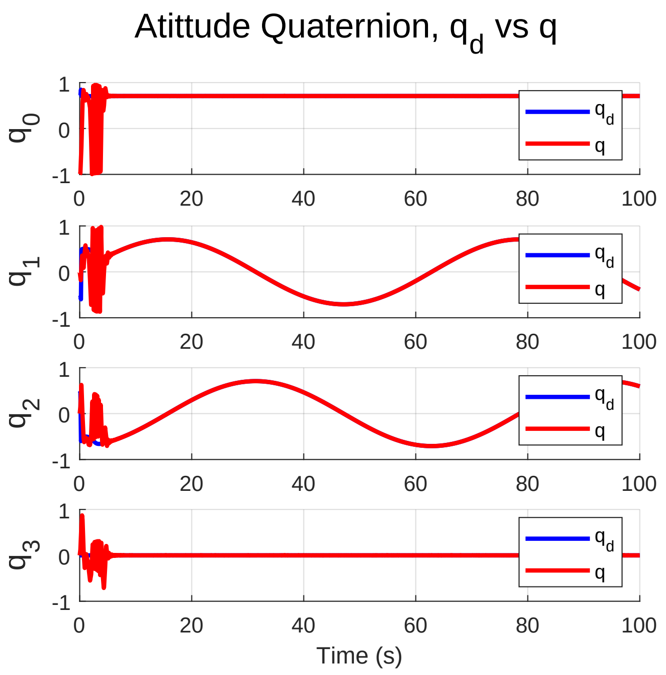


Figure 11: Desired quaternion ($\mathbf{q}_d$) vs Attitude quaternion ($\mathbf{q}$).

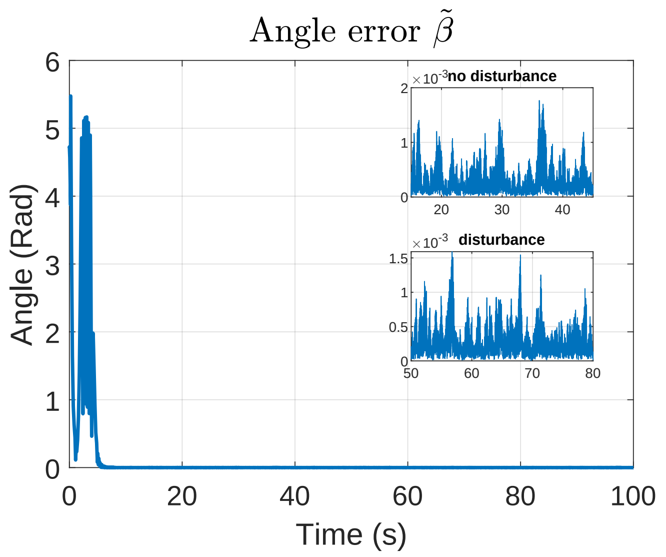


Figure 12: Error angle $\tilde{\beta}$.

ACKNOWLEDGMENTS

This work was made possible thanks to the CONACYT scholarship no. 546826 and the CONACYT Basic Science Project CB-2015-257985.

REFERENCES

[1] Gabe Hoffmann, Dev Gorur Rajnarayan, Steven L Waslander, David Dostal, Jung Soon Jang, and Claire J

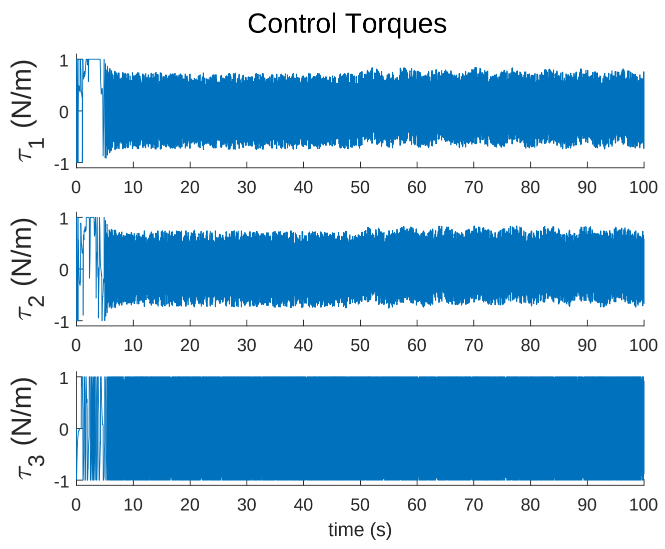


Figure 13: Control torques.

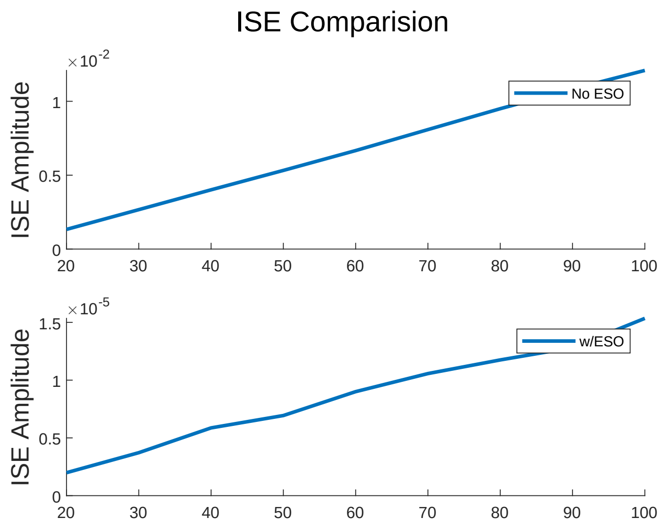


Figure 14: Control torques.

Tomlin. The stanford testbed of autonomous rotorcraft for multi agent control (starmac). In *The 23rd Digital Avionics Systems Conference (IEEE Cat. No. 04CH37576)*, volume 2, pages 12–E. IEEE, 2004.

[2] Minh-Duc Hua, Tarek Hamel, Pascal Morin, and Claude Samson. Introduction to feedback control of underactuated vtolvehicles: A review of basic control design ideas and principles. *IEEE Control Systems Magazine*, 33(1):61–75, 2013.

[3] Roberto Naldi, Michele Furci, Ricardo G. Sanfelice, and Lorenzo Marconi. Robust global trajectory tracking for underactuated vtol aerial vehicles using inner-outer loop control paradigms. *IEEE Transactions on Automatic Control*, 62(1):97–112, 2017.

[4] Ben Ammar Nour, Bouallegue Soufiene, and Haggege Joseph. Modeling and Sliding Mode Control of a Quadrotor Unmanned Aerial Vehicle. *3rd International Conference on Automation, Control, Engineering and Computer Science (ACECS)*, 3(1), 2016.

[5] G. Jithu and P. R. Jayasree. Quadrotor modelling and control. In *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pages 1167–1172, 2016.

[6] J. Fermi Guerrero-Castellanos, Nicolas Marchand, Ahmad Hably, Suzanne Lesecq, and Jérôme Delamare. Bounded attitude control of rigid bodies: Real-time experimentation to a quadrotor mini-helicopter. *Control Engineering Practice*, 19(8):790–797, 2011.

[7] H. Abaunza, P. Castillo, and R. Lozano. Quaternion Modeling and Control Approaches. *Handbook of Unmanned Aerial Vehicles*, pages 1–29, 2018.

[8] Cyrus Jahanchahi and Danilo P Mandic. A class of quaternion kalman filters. *IEEE Transactions on Neural Networks and Learning Systems*, 25(3):533–544, 2013.

[9] Erik B Dam, Martin Koch, and Martin Lillholm. *Quaternions, interpolation and animation*, volume 2. Citeseer, 1998.

[10] Gaopeng Bo, Liuyong Xin, Zhang Hui, and Wanglin Ling. Quadrotor helicopter attitude control using cascade pid. In *2016 Chinese Control and Decision Conference (CCDC)*, pages 5158–5163. IEEE, 2016.

[11] Elias Reyes-Valeria, Rogerio Enriquez-Caldera, Sergio Camacho-Lara, and Jose Guichard. Lqr control for a quadrotor using unit quaternions: Modeling and simulation. In *CONIELECOMP 2013, 23rd International Conference on Electronics, Communications and Computing*, pages 172–178. IEEE, 2013.

[12] Rafael Guardeño, Manuel J López, and Víctor M Sánchez. Mimo pid controller tuning method for quadrotor based on lqr/lqg theory. *Robotics*, 8(2):36, 2019.

[13] Samir Bouabdallah, Andre Noth, and Roland Siegwart. PID vs LQ control techniques applied to an indoor micro quadrotor. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2451–2456. IEEE, 2004.

[14] Shahida Khatoon, Dhiraj Gupta, and LK Das. PID & LQR control for a quadrotor: Modeling and simulation. In *2014 international conference on advances in computing, communications and informatics (ICACCI)*, pages 796–802. IEEE, 2014.

[15] Xing Huo, Mingyi Huo, and Hamid Reza Karimi. Attitude stabilization control of a quadrotor uav by using backstepping approach. *Mathematical Problems in Engineering*, 2014, 2014.

[16] Paul De Monte and Boris Lohmann. Trajectory tracking control for a quadrotor helicopter based on backstepping using a decoupling quaternion parametrization. In *21st Mediterranean Conference on Control and Automation*, pages 507–512. IEEE, 2013.

[17] An Honglei, Li Jie, Wang Jian, Wang Jianwen, and Ma Hongxu. Backstepping-based inverse optimal attitude control of quadrotor. *International Journal of Advanced Robotic Systems*, 10(5):223, 2013.

[18] Jihad Ghandour, Samir Aberkane, and Jean-Christophe Ponsart. Feedback linearization approach for standard and fault tolerant control: Application to a quadrotor uav testbed. *Journal of Physics: Conference Series*, 570(8):082003, 2014.

[19] Young-Cheol Choi and Hyo-Sung Ahn. Nonlinear control of quadrotor for point tracking: Actual implementation and experimental tests. *IEEE/ASME transactions on mechatronics*, 20(3):1179–1192, 2014.

[20] Kostas Alexis, Christos Papachristos, Roland Siegwart, and Anthony Tzes. Robust model predictive flight control of unmanned rotorcrafts. *Journal of Intelligent & Robotic Systems*, 81(3-4):443–469, 2016.

[21] Gaetano Tartaglione, Egidio D'Amato, Marco Ariola, Pierluigi Salvo Rossi, and Tor Arne Johansen. Model predictive control for a multi-body slung-load system. *Robotics and Autonomous Systems*, 92:1–11, 2017.

[22] Ahmed T Hafez, Anthony J Marasco, Sidney N Givigi, Mohamad Iskandarani, Shahram Yousefi, and Camille Alain Rabbath. Solving multi-uav dynamic encirclement via model predictive control. *IEEE Transactions on control systems technology*, 23(6):2251–2265, 2015.

[23] Hongwei Mo and Ghulam Farid. Nonlinear and adaptive intelligent control techniques for quadrotor uav–a survey. *Asian Journal of Control*, 21(2):989–1008, 2019.

[24] Yuanqing Xia and Mengyin Fu. *Compound control methodology for flight vehicles*, volume 438. Springer, 2013.

[25] Zeghlache SAMIR. Sliding mode control strategy for a 6 dof quadrotor helicopter. *Journal of Electrical Engineering*, 10(3):7–7, 2010.

[26] Xingling Shao, Jun Liu, Huiliang Cao, Chong Shen, and Honglun Wang. Robust dynamic surface trajectory tracking control for a quadrotor uav via extended state observer. *International Journal of Robust and Nonlinear Control*, 28(7):2700–2719, 2018.

[27] John Dixon. *Suspension Analysis and Computational Geometry*. Wiley, 2009.

[28] Malcolm D Shuster. A survey of attitude representations. *Navigation*, 8(9):439–517, 1993.

[29] Rune Schlanbusch, Antonio Loria, and Per Johan Nicklasson. On the stability and stabilization of quaternion equilibria of rigid bodies. *Automatica*, 48(12):3135–3141, 2012.

[30] Anežka Chovancová, Tomáš Fico, Ľuboš Chovanec, and Peter Hubinsk. Mathematical modelling and parameter identification of quadrotor (a survey). *Procedia Engineering*, 96:172–181, 2014.

[31] Bin Xian, Chen Diao, Bo Zhao, and Yao Zhang. Nonlinear robust output feedback tracking control of a quadrotor uav using quaternion representation. *Nonlinear Dynamics*, 79(4):2735–2752, 2015.

[32] Yong Zhang, Zengqiang Chen, and Mingwei Sun. Trajectory tracking control for a quadrotor unmanned aerial vehicle based on dynamic surface active disturbance rejection control. *Transactions of the Institute of Measurement and Control*, 42(12):2198–2205, 2020.

Design of aeroacoustically stealth MAV rotors

P. Li Volsi* and D. Gomez-Ariza
PARROT Drones SAS, Paris, 75010, France
R. Gojon, T. Jardin and J.-M. Moschetta
ISAE-SUPAERO, Université de Toulouse, Toulouse, 31400, France

ABSTRACT

The more restrictive airspace regulations force drone manufacturers to take into account the noise emitted during the design phase, along with the aerodynamic performance to increase the flight time. Here, a Non-Linear Vortex Lattice Method (NVLM), coupled with the Farassat Formulation-1A of the Ffowcs-Williams and Hawkings acoustic analogy is used to evaluate the aerodynamic and aeroacoustic performance of MAV rotors. Pymoo, a Python-based optimization framework, is employed to modify the geometry, evaluate its performance and extract the set of Pareto optimal solutions. The two objectives are the aerodynamic Figure-of-Merit and the Sound Pressure Level of the 1st Blade Passing Frequency peak for a microphone located in the rotor wake at a far-field distance of 1.62m and 30° from the rotor plane. The approach proposed in this paper takes into account up to ten different parameters, ranging from the twist and chord distributions, to the rake and skew angles.

1 INTRODUCTION

MAV drones are revolutionizing the world with their versatility and controllability. They are employed in the civilian sector by film producers, photographers and, in the near future, by enterprises for delivering goods in urban areas. They share the airspace with helicopters and airplanes but the operability costs and the dimensions of the latter make MAVs a real asset in difficult environments and urban areas (Figure 1 shows two drones used in civil and military sectors). For this reason, MAV usage is steadily increasing every year and they need to comply with new and more restrictive international regulations that include noise emissions and safety. Since in the future they will fly above people, the noise emitted should respect different criteria. Because of the high rotation speed of MAV propellers, their noise is unpleasant for the human hear and this is mainly caused by the highly coupled aerodynamic and aeroacoustic interactions between the rotors[1, 2], and with the drone body[3].

Figure 1: The Parrot ANAFI AI (on the left), the civil version of the Parrot ANAFI USA (on the right). Courtesy of Parrot Drones.

*Email address: pietro.livolsi@parrot.com

Another concern, this time for mission capabilities, comes from their low endurance. The viscous drag induced by the low Reynolds number, at which MAV rotors operate, reduces the aerodynamic efficiency of the rotors, hence reduces the endurance for a given energy storage.
The objective of this paper is to tackle both problems by providing an optimization framework that complies with industrial time and cost constraints. A Non-Linear Vortex Lattice Method, firstly introduced in sections 2.1 and 2.2, then validated in sections 2.3, is used in an optimization loop that exploits the multi-objectives properties of the python framework Pymoo (see section 3.1) to design acoustically and aerodynamically optimized rotors. The optimization results are presented and analyzed in section 4.

2 NUMERICAL METHOD

2.1 Non-Linear Vortex Lattice Method

The Non-Linear Vortex Lattice Method used in this work has been previously presented by Jo et al.[4, 5]. It is based on the incompressible ($\nabla \cdot \mathbf{V} = 0$), inviscid ($\nu = 0$) and irrotational ($\nabla \wedge \mathbf{V} = \mathbf{0}$) flow assumptions. The velocity vector is consequently expressed by the gradient of a potential flow ($\mathbf{V} = \nabla(\phi)$) that satisfies Laplace's equation: $\Delta(\phi) = 0$. These hypothesis allow to simulate complex flows by means of simpler potential flows. The blade is considered thin and its mean camber line is divided into $N_i \times N_j$ lattices whose vortex strengths are noted $\Gamma_{i,j}$.
The rotor wake is also modeled using vortex rings following a prescribed wake geometry. The sectional linear lift is obtained by applying the Kutta-Joukovsky theorem ($L_j = \rho_\infty V_\infty \Gamma_j$). A look-up table procedure is used to take into account the low Reynolds number induced non-linearity and is obtained by means of XFOIL[6] polar calculations. The approach is here used in a steady framework to reduce computational costs and comply with industrial constraints. However, it can be extended to unsteady simulations and free wake models, using free vortex particles to model the wake

as shown in Jo et al.[4, 5] (Figure 2 shows the two different simulations).

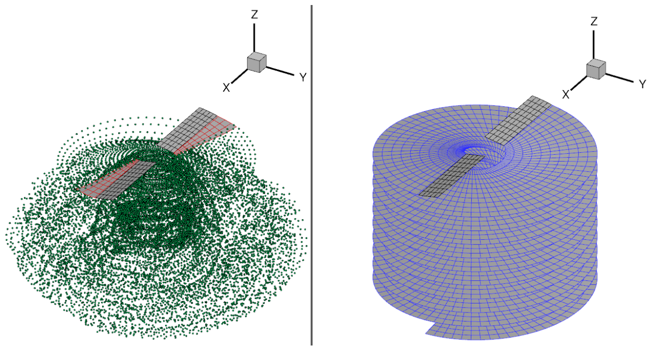


Figure 2: On the left, an unsteady simulation snapshot with Blade lattices, Wake Lattices (the red panels) and Vortex Particles (the green dots). On the right, a steady simulation snapshot with Blade lattices with the Prescribed Wake Lattices (the blue panels).

2.2 *Tonal noise - Farassat Formulation-1A*

The Formulation-1A presented by Farassat [7] has been implemented to the aforementioned Non-Linear Vortex Lattice Method code to capture the tonal noise spectrum emitted by the rotor. The tonal noise, for low-Reynolds and low-Mach number rotors, is generated by two sources:

$$p'(\mathbf{x},t) = p'_T(\mathbf{x},t) + p'_L(\mathbf{x},t) \tag{1}$$

- The thickness noise generated by the displacement of the fluid due to the blade passage, which depends purely on the blade geometry (through the $\mathbf{n}$ normal vector of equation 2) and the rotation speed (through the terms $\mathbf{v}$ and $\mathbf{M}$, that are respectively the absolute speed and the Mach speed of the elementary surface considered);

$$4\pi p'_T(\mathbf{x},t) = \int_{f=0} \left[\frac{\rho_0(\dot{v}_n + v_{\dot{n}})}{r\,|1-M_r|^2}\right]_{ret} dS + \int_{f=0} \left[\frac{\rho_0 v_n(r\dot{M}_r + cM_r - cM^2)}{r^2\,|1-M_r|^3}\right]_{ret} dS \tag{2}$$

- The loading noise that is dependent on the unsteady and steady pressure distributions on the blade ($\mathbf{l}$ and $\dot{\mathbf{l}}$ in equation 3) and the rotation speed:

$$4\pi p'_L(\mathbf{x},t) = \frac{1}{c}\int_{f=0} \left[\frac{\dot{l}_r}{r\,(1-M_r)^2}\right]_{ret} dS + \int_{f=0} \left[\frac{l_r - l_M}{r^2\,(1-M_r)^2}\right]_{ret} dS + \frac{1}{c}\int_{f=0} \left[\frac{l_r(r\dot{M}_r + c(M_r - M^2))}{r^2\,(1-M_r)^3}\right]_{ret} dS \tag{3}$$

$$\tau = t - \frac{r}{c} = t - \frac{|\mathbf{x}-\mathbf{y}|}{c} \tag{4}$$

With:

$$l_r = l_i\hat{r}_i \quad \dot{l}_r = \dot{l}_i\hat{r}_i \quad l_M = l_i\hat{M}_i$$

The observer position plays an important role through the $\mathbf{r}$ observer vector and the projection of the other vectors onto this one. Therefore, the inputs needed for the tonal noise calculations are: the observer position, the rotation speed, the pressure distribution on the mean camber line (calculated through the NVLM code and exclusively needed for the loading noise), and the 3D geometry (for the thickness noise).

2.3 *Validation*

To validate the code, unsteady simulations were compared with experiments from Gojon et al.[8] obtained on a two-bladed, NACA12-profiled, 10° constant pitch, constant chord rotor operating at 6000 RPM. Differences of +7.4% and +4.4% on thrust and torque coefficients have been obtained, respectively, which is within the uncertainty typically obtained between experiments with different test benches and different specimen of a given rotor geometry (Deters et al.[9], Gojon et al.[8]). Steady simulations provide comparable results with differences of 11% and 0.9% on thrust and torque coefficients respectively.
In addition, the acoustic simulations were compared to the experimental results of Gojon et al.[8]. As shown in Fig.3, differences lower than 3dB for microphones located in the rotor wake and between 3dB and 6dB for microphones located upstream of the rotor plane were observed. Since MAVs are usually flown above people, acoustic simulation for a microphone at 30° below the rotor is used for the optimizations of section 3. It was here verified that steady simulations provide similar sound pressure levels than unsteady simulations. Hence, because steady simulations are faster than unsteady ones by an order of magnitude, they are used in the optimization procedure to calculate the aerodynamic and acoustic performance of MAV rotors.

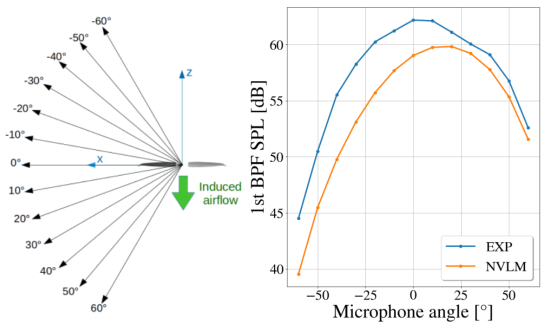


Figure 3: On the left, the microphone locations are displayed; On the right, the comparison between the experimental data and the simulations is shown.

3 OPTIMIZATION

The objective of this paper is to optimize the geometry of an MAV rotor in hovering conditions. The two optimization objectives are: The aerodynamic Figure-Of-Merit [10] $FM = \left(T^{3/2}\right) / \left(\omega Q \sqrt{2\rho\pi R^2}\right)$ and the Sound Pressure Level at the 1st BPF peak for a microphone located at a far-field distance of 1.62m and 30° below the rotor plane, in the direction of the flow. The ideal rotor has a higher Figure-of-Merit for an extended flight endurance and a lower SPL value to be acoustically stealthier.

Previous rotor aeroacoustic, multi-objectives optimizations [11, 12, 13, 14, 15, 16] have shown that, since these two objectives are in conflict, a single solution for this optimization that simultaneously improves both objectives does not exist. Instead, a series of optimal solutions that form a Pareto-front can be calculated. In this section, the optimization framework is presented (subsection 3.1), as long as the optimization algorithm in subsection 3.2, its parameters in subsection 3.3 and the design variables and constraints in subsection 3.4.

3.1 Pymoo package

The Python-native optimization framework Pymoo[17] is used in this work. This framework allows to make both single- and multi-objectives optimizations, based on different algorithms, like GA, CMAES, NSGA-II, etc. The reader is referred to reference [17] for further details.

3.2 Optimization algorithm

Genetic Algorithms (GA) are inspired by Charles Darwin's theory of natural evolution and based on the survival of the fittest, but also on the appearance of crossover combinations and mutations that can lead to fitter successive generations. Since they work with a population of solutions, different set of solutions can be maintained throughout the optimization and lead more easily to global minima, while gradient-based optimizations can get stuck to local ones. They can also identify the set of Pareto optimal solutions. The main difference between GAs lies in the survival and selection methods.
The NSGA-II (Non-dominated Sorting Genetic Algorithm) [18] was chosen for this study. This algorithm is one of the first genetic algorithm and was opportunely modified from the first version of the code to improve the convergence speed and use the elitism as a way to increase the performance and prevent the loss of good solutions.

3.3 Optimization setup

As highlighted in the previous subsection, genetic algorithms like the NSGA-II used in this study require the number of candidates that will be part of the initial population. In this case, a population of 100 candidates has been chosen. The candidates are randomly chosen in order to increase the diversity and the possibility of finding good candidates. The selection is made through the tournament selection method that selects a number of individuals, compares their fitness and selects the parents that will then be modified through the crossover operations and mutations to give birth to the new generation.
The crossover operation used is the "Simulated Binary Crossover" (more details on [19]). This operation, also called recombination, combines the genetic data of different parents to create a newborn.
The mutation operation, instead, randomly modifies the parameters by taking into account a given probability. The mutation probability is here set to zero.

3.4 Design variables, geometrical reconstruction and constraints

The rotor geometry on which the optimizations are based is described in table 1:

Rotor parameters	
# of blades	2
Airfoil Section	NACA0012
Diameter [m]	0.25
Root cut-out	15% of the Radius

Table 1: Non-optimized rotor parameters.

The design variables taken into account in the optimization are listed in table 2 along with the minimum and maximum bounds.

Design variables			**Min Value**	**Max Value**
Twist	CP_{twist}	Angle	5°	30°
		Position	20%	80%
	TIP_{twist}	Angle	0°	10°
Chord	CP_{chord}	Length	0.025m	0.075m
		Position	20%	80%
	TIP_{chord}	Length	0.005m	0.05m
Skew	CP_{skew}	Position	20%	80%
	TIP_{skew}	Angle	-10°	10°
Rake	CP_{rake}	Position	20%	80%
	TIP_{rake}	Angle	-10°	10°

Table 2: Design variables used in this work and their minimum and maximum bounds.

A continuous function allows the optimization algorithm to choose the values of each parameter within the minimum and maximum bounds (the number of possible combinations is set by machine precision). Once the ten variables are calculated by the optimization algorithm, the NVLM code builds the new geometry. The chord and twist distributions are defined as follow:

- The root chord and pitch are respectively fixed at 0.025m and 10°;

- The CP$_{chord,pos}$ defines the spanwise position at which the CP$_{chord,len}$ is applied. The derivative at this point is fixed to zero (same strategy for the twist distribution with CP$_{twist,pos}$ and CP$_{twist,ang}$ variables);
- The chord length at the tip of the blade is defined by TIP$_{chord,len}$ (the pitch at the tip by TIP$_{twist,ang}$).

Once the interpolation function is defined by the previous points, the geometry is interpolated and 10 spanwise values of the twist angles and the chord lengths are calculated to generate the geometry.
A similar approach for the skew and rake (commonly known as winglet) distributions has been used:

- Both skew and rake angles and their derivatives are equal to zero at the root of the blade;
- The TIP$_{skew,ang}$ and TIP$_{rake,ang}$ variables define the angles at the tip;
- The CP$_{skew,pos}$ is the last point (going from the root to the tip of the blade) to have both skew/rake angles and their derivatives equal to zero.

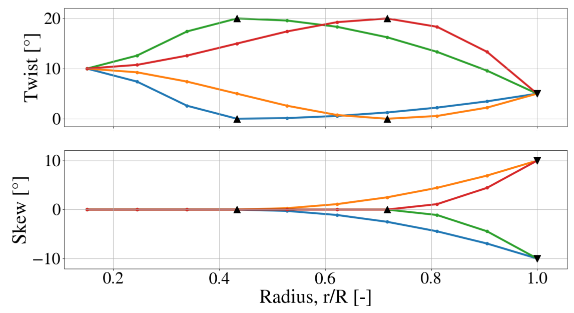


Figure 4: Twist (on top) and skew (on the bottom) distribution definition from the Control point (the black upward pointing triangles) and the tip value (the black downward pointing triangles).

Figure 4 shows, on top, an example of twist distribution interpolation. The upward pointing triangles represent the four control points with two different positions and two different values of the pitch angle, while the downward pointing triangles represent the tip value (for the sake of visibility it was kept constant in the plot). On the bottom an example of skew/rake distribution is presented: in this case both control points (represented by the upward pointing triangles) and angles at the tip (the downward pointing triangles) are changed and 4 different distributions are created.
The NACA0012 airfoil has been kept constant to limit the size of the parameter space.

To optimize both endurance and noise of a ∼800 grams MAV drone, the MAV single-rotor optimization is conducted at 2N iso-thrust. To obtain this thrust value the following procedure has been put in place:

- The algorithm chooses the values of the design parameters and the i^{th} geometry is generated;
- An aerodynamic calculation at 4000RPM is run and the mean thrust calculated;
- By making the assumption that the thrust follows an ideal quadratic function (Thrust $= a \cdot \Omega^2$), the ”a” coefficient is calculated and the rotation speed Ω_{2N} deduced[1].
- A new aerodynamic calculation is run and, with it, an acoustic one in order to get the two objective values FM and 1st BPF SPL.
- The two objective values are evaluated by the NSGA-II algorithm.

In the following section, the results of four different optimizations are presented: one optimization does not take into account the rake/skew angles, one takes into account the rake distribution (but not the skew), while another one the skew (but not the rake), and one takes both into account. This helps assess the role of rake and skew on optimal solutions.

4 Results

In this section the results from the aerodynamic and aeroacoustic optimizations of MAV rotors using the genetic algorithm NSGA-II and the non-linear vortex lattice method coupled with the Farassat Formulation-1A solution of the Ffowcs-Williams and Hawkings analogy are presented. As explained in the previous section, two calculations per iteration are needed, which means that for a total of 40 generations (with 100 candidates each), 8000 simulations need to be computed. For this reason, having a low computational cost per simulation is crucial and, therefore, the steady-simulation feature has been preferred over the unsteady counterpart. The NVLM main parameters are summed-up in table 3.

Blade discretization	
# of Chordwise Lattices (N_i)	5
# of Spanwise Lattices (N_j)	10
Simulation parameters	
Revolutions [-]	5
Step angle [°]	5

Table 3: Simulation parameters used for the validation of the NVLM code.

Figure 5 shows the Pareto-fronts obtained with four different optimization setups, including or not rake and skew distributions in the design variables.

[1]within 6% of the target thrust.

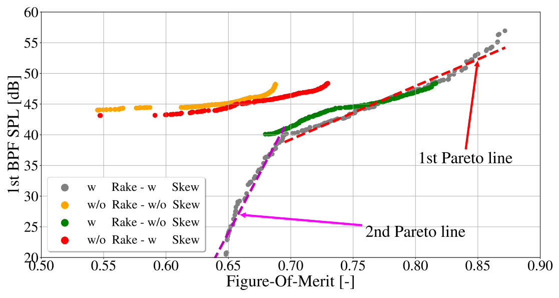


Figure 5: Scatter plot with four different Pareto-fronts and the two Pareto lines.

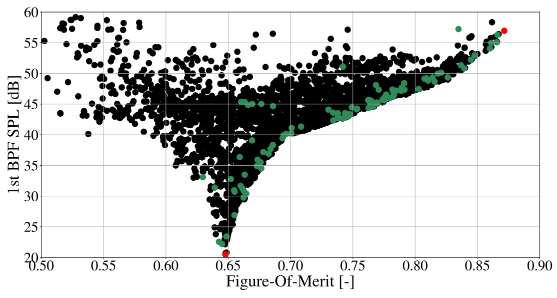


Figure 6: Scatter plot representing all candidates simulated: the last generation candidates in green and the best aerodynamic efficient and the acoustically stealthier candidates in red (whose geometries are shown in figure 12).

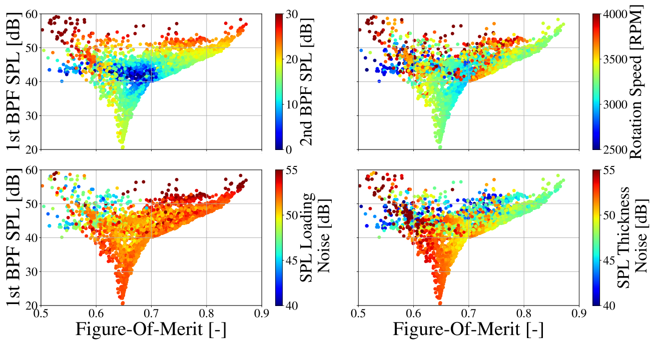


Figure 7: Four scatter plots showing the candidates coloured by different variables.

The orange-colored Pareto-front represents an optimization with only 6 design variables, since it does not include skew and rake variables into the optimization loop. Red and green Pareto-fronts are 8-variables optimizations: they include the skew and rake distributions in the rotor design respectively. This allows to slightly improve the performance of the Pareto set of optimal rotors. It is shown that including the skew variables helps to move the Pareto-front towards better aerodynamic efficiencies but does not yield significant changes in acoustic performance. When the rake variables are introduced in the loop, instead, the optimization seems more appropriate to find aerodynamically ideal rotors and acoustically stealthier ones with 5dB difference with respect to the orange curve. The last optimization, depicted with the grey dots, includes both rake and skew distributions into the design parameters and has the widest Pareto-front, with both aerodynamic and acoustic performance overtaking the ones given by the green-colored rake-included Pareto-front. It also shows two linear trends: the red one (referred to as ”1st Pareto line”) and the magenta one (”2nd Pareto line”). For this reason, only this last optimization run will be presented and analyzed in more details.

The dots of figure 6 scatter plot represent all the candidates of the 10-variables optimization. The green ones, that represent the last generation candidates, are all very close to the Pareto-front and show that the optimization algorithm has converged to the Pareto set of optimal solutions.

As described in subsection 2.2, the total tonal noise depends on two sources (the loading noise and the thickness noise). For this reason it is important to analyze the interaction between them to understand the previous graph. Figure 7 shows four differently coloured Pareto-fronts.

The main results are:

- The rotation speed (figure 7 - top-right scatter plot) changes along the Pareto-front. By following the 1st Pareto line from right to left, the rotation speed increases until the junction between the two Pareto lines. Here, the rotation speed drops suddenly, to increase again along the 2nd Pareto line. While the influence of a reduced rotation speed on the reduction of tonal noise is explicit from equations 3 and 2, these results indicate that it may be counterbalanced by other terms such as rotor solidity (shown in figure 10 and defined as follow: $\sigma = \left(N_{blades} \cdot \overline{chord}\right) / (\pi R)$);
- The thickness noise (figure 7 - bottom-right scatter plot) follows the same trend of the rotation speed. A steady increase from right to left of the first Pareto line is followed by a sudden decrease in the thickness noise level and, again, an increase along the second Pareto line;
- At a far-field microphone location, the loading noise (figure 7 - bottom-left scatter plot) depends mainly on the magnitude of thrust force. However, even for geometries having the same thrust, differences in the order of 3dB are depicted in the graph and depend on the loading distribution, but also on the rotation speed of the rotor (see equation 3);
- The geometries colored by the 2nd BPF SPL (figure 7 - top-left scatter plot) show that optimizing the 1st BPF does not affect, in the same way, the 2nd BPF.

Overall, it can be understood from these results that while loading noise dominates the acoustic footprint on the first Pareto line, it is competed by thickness noise on the second Pareto line. An increase in the rotation speed can be compensated for by an increase in rotor solidity to achieve a given target thrust. Furthermore, an increase in rotor solidity increases thickness noise through its explicit contribution in equation 2. Because the thickness noise is weak compared to the loading noise on the first Pareto line, an increase in rotor solidity that significantly impacts the SPL and a reduction in SPL can be directly correlated with a decrease in RPM. Conversely, because the thickness noise is of the same order of magnitude than loading noise on the second Pareto line, an increase in rotor solidity contributes to an increase in SPL (see figure 10). Hence SPL is not solely correlated with RPM, which explains why rotor geometries with lower acoustic footprint may not be obtained at minimum rotation speeds. However, a closer look at the 2nd BPF peaks shows how improving the 1st BPF peak does not mean a consequently improvement of the 2nd one as well.

Figures 8 and 9 show the chord lengths, as well as the pitch, rake and skew angles of the different Control Points (CP) and the Tip (TIP). Here are additional insights on the Pareto optimal solutions:

- The chord at the control point reaches the maximum value allowed to the algorithm. This maximum value is located near the root (see top-left scatter plot of figure 11);
- The chord at the tip (figure 8 - top-right scatter plot) is the key parameter to understand the two Pareto lines: the geometries with higher Figure-Of-Merit have a smaller chord at the tip, which contributes to reduce tip vortex strength. The right Pareto line, instead, presents larger chord lengths at the tip to increase rotor solidity, which may affect thickness/loading noise cancellation mechanisms;
- Rotors with larger aerodynamic efficiency have a higher pitch angle at the control point (see figure 8 - bottom-left scatter plot) because they need to recover the thrust force that is lost by the smaller chord at the tip. The control points are located near the root (see figure 11);
- The pitch at the tip (figure 8 - bottom-right scatter plot) is almost constant everywhere, except on the left side of the first Pareto line. Here, in fact, a higher rotation speed compensates for the lower twist at the tip;
- The rake value is almost constant for all the geometries of the Pareto-front. It points downward, reaches the maximum value and is located very close to the tip. This probably has a double effect: it helps to reduce the torque induced by the tip vortex, and add an outward component to the force vector that changes the phase of both loading and thickness noise, thus reducing the total noise levels;
- The Pareto-front shows two different and opposite skew angles, but the sudden change does not happen on the junction between the two Pareto lines, it happens on the first Pareto line.

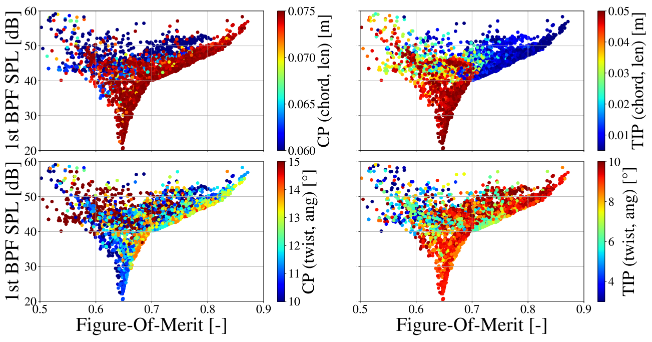


Figure 8: Four scatter plots showing the candidates coloured by design variable values.

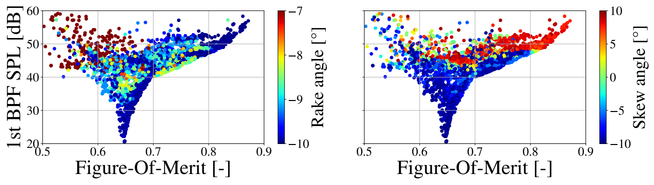


Figure 9: Two scatter plots showing the candidates coloured by rake angle (on the left) and skew angle (on the right) at the tip.

This optimization run shows how three design parameters are important to the improvement of the performance of MAV rotors. The chord length at the tip is the key parameter to understand the two Pareto lines. A smaller chord at the tip can lead to aerodynamically more efficient rotors. Larger ones to acoustically stealthier rotors. The rake and skew distributions play important roles in both aerodynamic and acoustic performance. The former, inducing a downward-pointing blade tip, reaches the maximum values allowed to the algorithm. The skew, instead, deforming the blade in the forward direction for acoustically stealthier rotors and backward for aerodynamically more efficient rotors. The two best geometries are shown in figure 12.

5 CONCLUSION

The approach here presented makes use of a non-linear vortex lattice method, coupled with Farassat's aeroacoustic tonal noise model and the genetic algorithm NSGA-II of

Pymoo package to optimize the rotor geometries and find aerodynamically more efficient and aeroacoustically stealthier MAV rotors. The two optimization objectives chosen for this study are the Figure-Of-Merit describing the rotor aerodynamic efficiency and the 1st BPF SPL peak for a microphone located at a far-field distance of 1.62m, at an angle below the rotor plane of 30°. Since drones are generally flown over populated areas and kept at a safety horizontal distance from people, this value of the angle is reasonable. However, the optimization does not take into consideration other angles and, therefore, the influence of the microphone angle will be assessed on future works.

All the optimization runs take into account the twist and chord distributions in the generation of new geometries (the airfoil chosen is the NACA0012), for a total of six design variables. Two optimizations with two additional variables, namely the rake and skew, are independently added. A last optimization run, including all ten design variables, has been performed and has further improved the set of optimal solutions.

The combined effects of both rake and skew on aerodynamic and aeroacoustic performance allowed reaching more efficient and stealthier rotors. The Pareto-front presents two linear trends with different slopes, referred to as Pareto lines. The parameter that splits the two lines is the chord length at the tip. A smaller chord, in fact, is preferable for aerodynamically more efficient rotors while a larger chord gives stealthier rotors.

The negative rake concentrated at the tip pushes the whole Pareto-front to aerodynamically more efficient rotors. Coupling the skew modifications to the rake ones, creates rotors with even higher Figure-Of-Merit and allows to go further on the acoustic counterpart too.

The 1st BPF SPL is only a part of the acoustic spectrum. The acoustic improvements here obtained do not imply improvements over the entire frequency spectrum. In fact, the 2nd BPF peak of the best acoustic configuration is not the lowest value obtained. In addition, if the sensitivity of the human ear is to be considered, the A-ponderation should be applied.

These results are obtained by using the steady simulation capabilities of the NVLM code. However, to validate all the presented results, experimental tests are being prepared at ISAE-SUPAERO. It is also worth noting that this optimization does not take into account the inertia of the blades, the rotation speed at which the motor is more efficient, and the rotor mass. These parameters must be taken into account into future optimizations of commercial MAV rotors.

ACKNOWLEDGEMENTS

This work has been financed by Parrot Drones SAS and the Association Nationale Recherche Technologie (ANRT), realized at ISAE-SUPAERO and performed using HPC resources from GENCI-IDRIS and GENCI-CINES on Jean Zay, Occigen (Grant A0102A07178) and CALMIP on Olympe (Grant 2021-p1425). The authors thank Joseph Morlier and Michael Bauerheim for their insights and advices.

REFERENCES

[1] H. Lee and D. J. Lee. Rotor interactional effects on aerodynamic and noise characteristics of a small multirotor unmanned aerial vehicle. *Physics of Fluids*, 32(4), 2020.

[2] E. J. Alvarez, A. Schenk, T. Critchfield, and A. Ning. Rotor-on-Rotor Aeroacoustic Interactions of Multirotor in Hover. *Journal of the American Helicopter Society*, 2020.

[3] N. S. Zawodny and D. D. Boyd. Investigation of rotor-airframe interaction noise associated with small-scale rotary-wing unmanned aircraft systems. *Annual Forum Proceedings - AHS International*, pages 66–82, 2017.

[4] Y. Jo, H. Lee, and D. J. Lee. Prediction of rotor flow for unmanned aerial system using nonlinear vortex lattice method. *6th Asian-Australian Rotorcraft Forum and Heli Japan 2017, ARF 2017*, 2017.

[5] Y. Jo, T. Jardin, R. Gojon, M. C. Jacob, and J.-M. Moschetta. *Prediction of Noise from Low Reynolds Number Rotors with Different Number of Blades using a Non-Linear Vortex Lattice Method.*

[6] M. Drela. Xfoil: An analysis and design system for low reynolds number airfoils. In Thomas J. Mueller, editor, *Low Reynolds Number Aerodynamics*, pages 1–12, Berlin, Heidelberg, 1989. Springer Berlin Heidelberg.

[7] F. Farassat. Linear acoustic formulas for calculation of rotating blade noise. *AIAA Journal*, 19(9):1122–1130, 1981.

[8] R. Gojon, T. Jardin, and H. Parisot-Dupuis. Experimental investigation of low reynolds number rotor noise. *The Journal of the Acoustical Society of America*, 149(6):3813–3829, 2021.

[9] R. W. Deters, G. K. Ananda, and M. S. Selig. Reynolds number effects on the performance of ailerons and spoilers. *39th Aerospace Sciences Meeting and Exhibit*, (June):1–43, 2001.

[10] G. J. Leishman. *Principles of helicopter aerodynamics / J. Gordon Leishman,...* Cambridge aerospace series. Cambridge University Press, New York, second edition edition, right 2006.

[11] R. Serré, N. Gourdain, T. Jardin, G. Delattre, and J.-M. Moschetta. Analysis of the flow produced by a low-Reynolds rotor optimized for low noise applications. Part II: Acoustics. *43rd European Rotorcraft Forum, ERF 2017*, 2:799–805, 2017.

[12] R. Serré, N. Gourdain, T. Jardin, M. C. Jacob, and J.-M. Moschetta. Towards silent micro-air vehicles: optimization of a low Reynolds number rotor in hover. *International Journal of Aeroacoustics*, 18(8):690–710, 2019.

[13] C. Nana, M. Yann, and R. Serré. Fast Multidisciplinary Optimization of a MAV propeller for noise reduction: from simulation to experimentation. *53rd 3AF International Conference on Applied Aerodynamics*, (June), 2018.

[14] C. F. Wisniewski, A. R. Byerley, W. H. Heiser, K. W. Van Treuren, and W. R. Liller. Designing small propellers for optimum efficiency and low noise footprint. *33rd AIAA Applied Aerodynamics Conference*, (June):1–17, 2015.

[15] F. Boyer and A. Drapier. Multidisciplinary optimization of a MAV propeller for noise reduction. pages 301–306, 2017.

[16] G. Wilke. Findings in aero-acoustic simulations for optimizations. In *76th Annual Forum*, Virtual, October 2020.

[17] J. Blank and K. Deb. Pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.

[18] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[19] K. Deb, K. Sindhya, and T. Okabe. Self-adaptive simulated binary crossover for real-parameter optimization. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, GECCO '07, page 11871194, New York, NY, USA, 2007. Association for Computing Machinery.

APPENDIX A: RESULTS AND GEOMETRIES

In this appendix three images are presented:

- A scatter plot showing the candidates colored by rotor solidity (figure 10);
- Four scatter plots in figure 11 coloured by the following variables: relative spanwise position of the chord control point ($CP_{chord,pos}$ at top-left), relative spanwise position of the twist control point ($CP_{twist,pos}$ at top-right), relative spanwise position of the rake control point ($CP_{rake,pos}$ at bottom-left) and relative spanwise position of the skew control point ($CP_{skew,pos}$ at bottom-right);
- Three different views (From top to bottom: Z, Y and 3D view) of figure 12 showing two rotors corresponding to the red dots of figure 6.

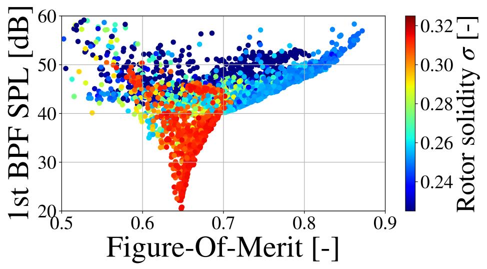


Figure 10: Scatter plot showing the candidates colored by rotor solidity.

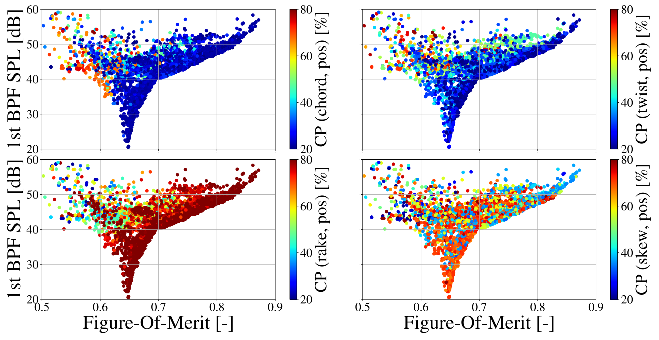


Figure 11: Scatter plot showing the candidates coloured by the control point variables.

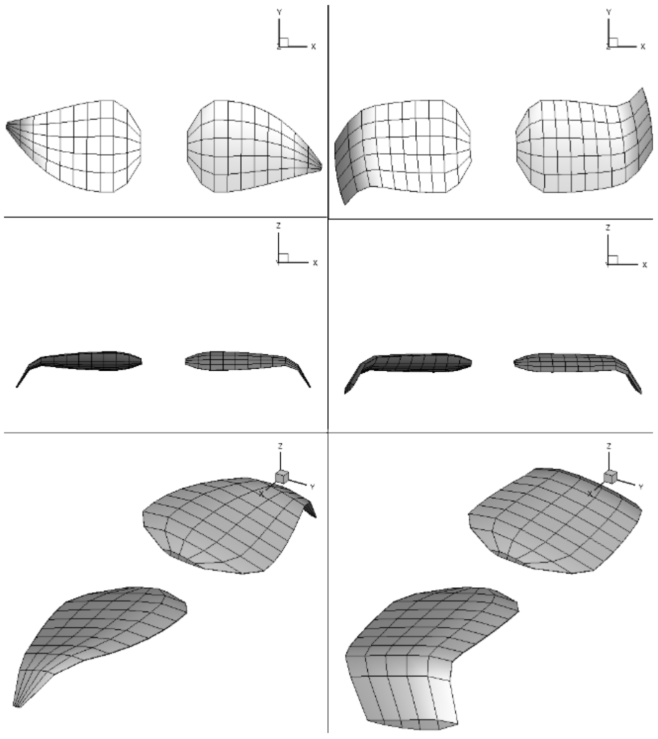


Figure 12: On the left: the rotor with highest Figure-Of-Merit, on the right: with the lowest acoustic footprint.

A click mechanism moderates drone's flapping wing kinematics for enhanced thrust generation

Shang-Lin Wu [1], Yao-Wei Chin[2], and Gih-Keong Lau[1]
National Yang Ming Chiao Tung University, 1001 University Road, Hsinchu, Taiwan
National University of Singapore, 21 Lower Kent Ridge Road, Singapore

ABSTRACT

In the motorized drive of a flapping-wing drone, the transmission loss well exceeds the inertial and aerodynamics power. Such transmission loss is however minor in the natural flight apparatus of Dipteran insects by using thoracic compliant mechanisms. In particular, Dipteran insects such as fruit flies make good use of a bi-stable click mechanism to enhance thrust generation and efficiency. It is not clear if an enlarged click mechanism remains beneficial to a larger flapping-wing drone. Here, we designed a bioinspired drone prototype, with a 40 times larger wingspan and a 300 times heavier mass than a fruit fly of 7.3mm span and 0.1gram mass. Interestingly, given the same wing pairs, the enlarged click mechanism enhanced the thrust generation (up to 30 grams) by 50% than the non-click mechanism. This click mechanism modulated the wing stroke speed profile to have a high plateau at which the wing rotation peaks. This insight on the click effect on wing kinematics and aerodynamics will help better future bioinspired drones.

1 INTRODUCTION

Natural flyers like birds and insects have been the source of inspiration for the development of bioinspired micro air vehicles (MAV), or flapping-wing drones in other words. Bird-inspired drones were named ornithopters, while insect-inspired drones were named entomopter. Artificial flapping-wing drones had exploited the bio-inspired aerodynamics, such as clap-and-fling wing-wing interaction employed by wasps and even pigeons during the laters' vertical takeoff. However, the drive mechanism for flapping-wing drone is still motorized transmission, for example, a crank-rocker mechanism, which converts the motor rotation into wing reciprocation. A recent study shows the transmission loss can overweigh the inertial and aerodynamics power expenditure during flapping-wing drive. This challenge motivates us to relook into the insect drive mechanism for a simpler and better design of the flapping wing mechanism [1].

Click mechanism are first found on Diptera flies while anesthetized [2][3]. The flight thorax on dipterans is more than just a box-like structure; it can store elastic energy when deformed and release when elastically recoiled. Such means of elastic energy storage were reported to help decelerate and accelerate wings at reduced inertial power expenditure. It was thought to be beneficial to increase the power output more than what flight muscles alone can be delivered. In addition, the click mechanism found at the wing base of Dipteran insects was thought as useful as deformable flight thorax for elastic storage.

Brennan (2003) simplified the click mechanism as several link-spring models as shown in figure 1 [4][5]. When the tip-point is pushed upward, the spring is compressed and generates a vertical force against the moving direction. After the tip-point

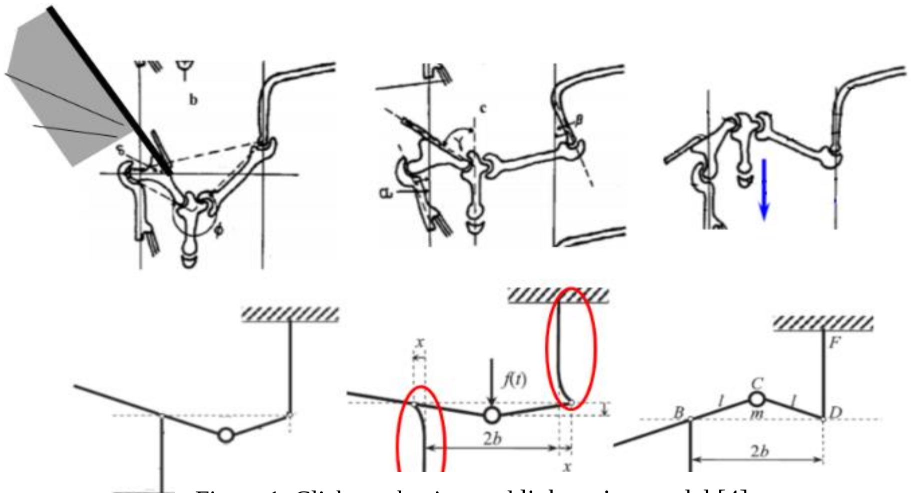


Figure 1: Click mechanism and link-spring model [4].

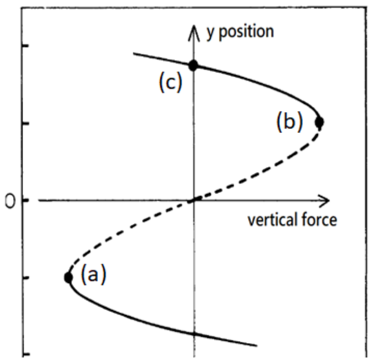


Figure 2: Vertical force versus position of click mechanism [5].

*Email address(es): wushanglin.c@nycu.edu.tw, tslcyw@nus.edu.sg, mgklau@nctu.edu.tw passes the mid-line, the vertical force from the spring turns to the same direction as motion, triggers the "click," and accelerates the tip-point to jump to the other side.

Chin (2013) designed a 3.78g weighted ornithopter prototype of a click mechanism for flapping a pair of 130mm-span wings, as shown in figure 4. Chin's works show that the click mechanism effectively produces higher thrust than a non-click prototype at both the same flapping frequency and power consumed [6][7]. However, the wing kinematics delivered by the motorized click transmission (a combination of click mechanism at the wing base and the driving crank-slider mechanism) appeared as a distorted sine wave rather than the bi-stable snaps demonstrated by the click mechanism alone. Further, it is noted the elastic storage in the click mechanism is maximum at the mid-stroke of flapping wings; it is in contrary to the maximum elastic energy storage expected at the end of wing stroke. His later research showed partial elastic energy storage in the elastic hinged wings is better than full elastic energy storage for not impeding motorized flapping-wing transmission.

It was not clear if the click mechanism found in Dipteran insect is applicable to a larger flapping-wing drone in the same manner as to how the clap-and-flying wing-wing interaction scales from small wasp to larger pigeons. In this paper, we first enlarged a click mechanism for a 27.6-gram flapping-wing drone of 300 mm wingspan. In addition, we find the effect of click mechanism moderating the wing kinematics distinctly at relatively low frequency (4Hz) and high frequency (10Hz).

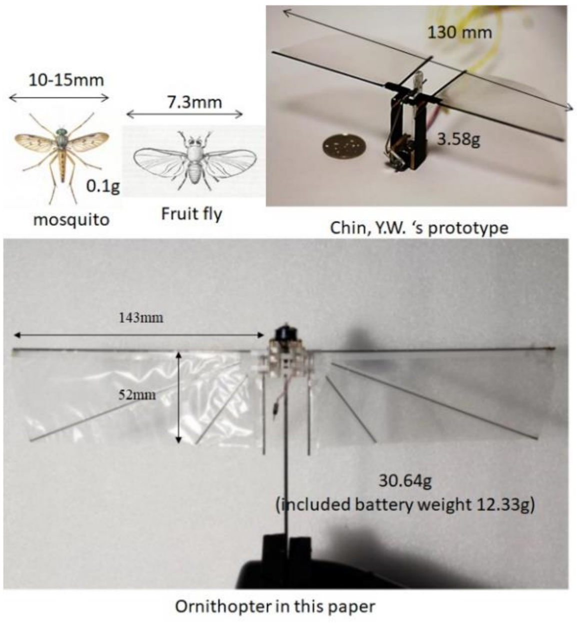


Figure 3: Size of Dipteran, Chin's prototype and ornithopter in this paper.

2 MECHANISM MODEL

2.1 Mechanism model design

We designed a large click ornithopter with two 14.3mm-long wings at a stroke of 106 °. This click mechanism is similar to that presented by Tang and Brennan but differs from the latter by the drive mechanism. Here, a crank-rocker mechanism was to pull down and push up the vertex of the click mechanism. A powerful small brushless motor AP05 and a set of speed-reduction gear sets were used to drive the crank-rocker mechanism. As in figure 4, the driving force is transmitted through reduction gear sets, a crank-rocker system (red linkages) and a double rocker system (black linkages) from the motor to wings. While in the "click" case, a cantilever spring replaces one of the rockers in the double rocker system.

As such, the wing kinematics follow the reciprocation primarily by the crank-rocker mechanism. Meanwhile, the click mechanism moderated the harmonic wing stroke profile by providing extra elastic resistance towards the mid-stroke position and extra elastic recoil away from the mid-stroke position.

2.2 Pull test

The spring of the click structure is a POM cantilever beam, as shown in figure 5. While the black links moved toward the mid-point, the cantilever spring was pressed toward the left and exerted force (red arrow) to the wing base. Thus we do the pull test to plot the vertical reaction force.

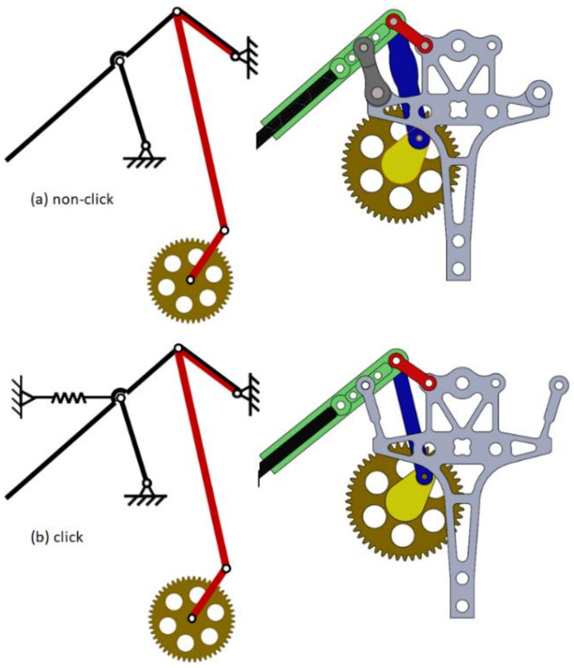


Figure 4: Transmission mechanisms. Crank (yellow): 4mm, link (blue): 15.1mm, rocker (red): 5mm, wing base (green): 5mm, rocker of non-click (gray): 6mm.

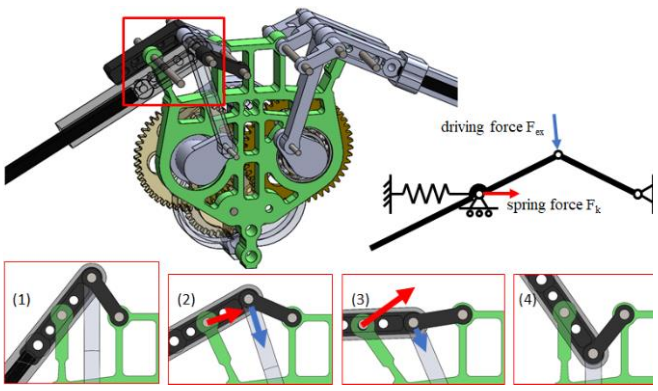


Figure 5: Click mechanism in this paper. The left green beam is the cantilever spring.

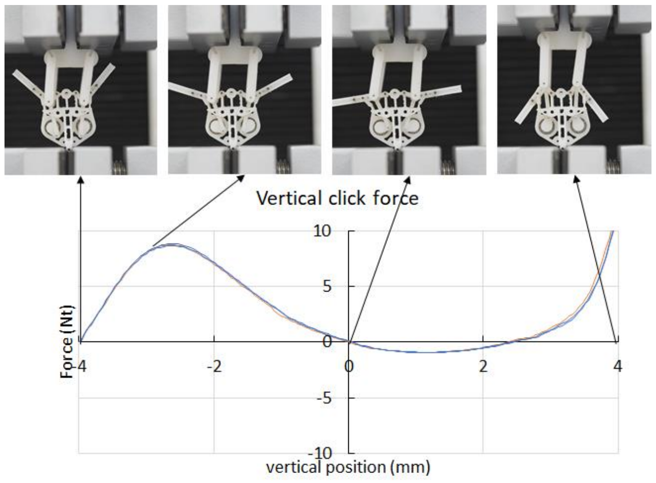


Figure 6: Vertical force tested with tensile test machine.

As shown in Figure 6, the bi-stable elastic behavior of a prototype of click mechanism was measured from a pull test using a tensile tester (Cometech QC-508M1F). Prior to the pull-test, the vertex of the click mechanism was set to the bottom stable position. During the test, the vertex was pulled upwards from the bottom stable position -4mm to the top stable position +4m while the pulling force was measured continuously. The pulling force required at the bottom stable position was zero; it increased towards a peak pull force of 8.7N at -2.6mm. When pulled up beyond the peak force position, the force requirement diminishes towards and becomes zero at the mid-stroke position.

Snap happens beyond the mid-stroke position with the pull force turning negative until the 2.5mm position. There was increasing positive pull force required to reach the top stable position because of misalignment in the pulling rod. Backlashes and skewed jigs are the reason for the asymmetry of the curve.

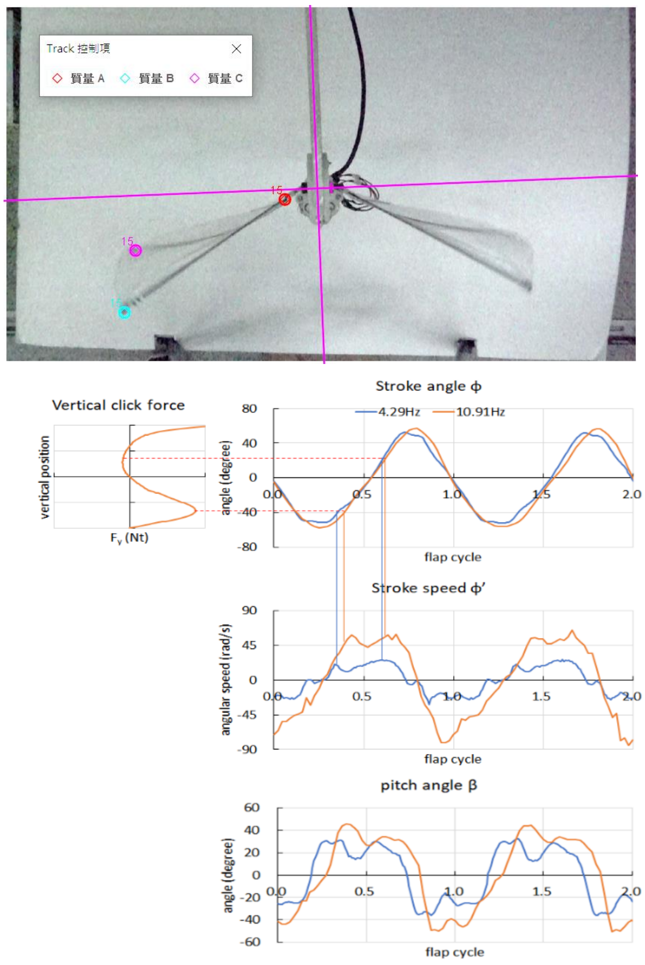


Figure 7: Stroke angle, Stroke speed, and pitch angle versus time.

3 Wing Kinematics

Next, we investigated the dynamics effect of the click mechanism on flapping-wing kinematics at low and high frequencies. First, the brushless motor was run at 40% throttle to beat a pair of wings hinged on a clicking mechanism at a low frequency of 4.29Hz. Second, the motor was driven to full throttle and flap the wing pair faster at a wingbeat frequency of 10.91Hz. During the test, a high-speed camera was used to record the wing motions. Subsequently, Tracker software was used to extract the stroke angle and pitch angle over cycles of wingbeat.

Figure 7 shows that the wing kinematics over two wingbeat cycles. It noted that the time profile of wing stroke angle appeared like a sinusoidal function, following the drive of the crank-rocker mechanism. However, the stroke amplitude varies with the wingbeat frequency. For example, the stroke amplitude was 57.1 ° at 10.91Hz wingbeat frequency; it is higher than 52.8° at 4.29Hz. The stroke amplitude at full throttle was amplified more than static stroke because of the elasticity provided by the click mechanism to the wing base.

Stroke speed can be calculated as the time derivates of stroke position. As shown in Figure 7, the time profile of stroke speed measured deviates from a simple harmonic function. Noted was a speed moderation, i.e., a mild dip, at the mid-stroke position, where a peak stroke speed was expected. The speed dip and the maximum spring resistance happened in the same phase. In other words, the resistive spring force of click slowed down the wingbeat and depressed the stroke speed curve at the mid-stroke position. After the wings pass the middle stroke, the recoiled spring force helped accelerate the wingbeat and keep the 'plateaus' of relatively high speed for a longer duration. The matching of high stroke speed with a high pitch angle enhances the thrust generation. As such, the stroke speed moderation is believed helpful to enhance thrust generation at this moderate wingbeat frequency of 10Hz.

4 FLIGHT TEST

Next, we conducted a fixed flapping test to measure the maximum thrust of this click-ornithopter prototype with a pair of wings each of 143mm wing length and 52mm chord width. As shown in Figure 8, a 6-axis force/torque sensor was used to measure the thrust generated by the flapping-wing prototype mounted below it. It is shown that ornithopter with click generates 30.83g thrust while non-click ornithopter only generates 22.20g. The click mechanism increased the maximum throttle by 35.7% and remained the same power consumption.

Last but not least, this click-ornithopter was tested for a free vertical takeoff. During the launch test, the click-ornithopter's fuselage, i.e., a carbon fiber rod body, was guided in a PVC tube. A tail stabilizer was stripped off the prototype but replaced with an extra weight of 3gram to enable free takeoff from the guiding tube. Figure 10 shows the initial state and the launched state of the click-ornithopter. The tethered flight test showed that this prototype was capable of 30.06-gram average thrust generation when driven at full throttle.

5 CONCLUSION

In conclusion, the click mechanism, which alone was bi-stable at the end stroke position, did not snap the wings pass mid-stroke position when it was under the drive by a motorized crank-rocker mechanism. However, it presented extra resistive elastic load to alter the stroke speed of flapping wings. Its elastic resistance and recoil help widen and smoothen the plateau of relatively high speed nearly the mid-stroke position where the pitch angle was high. As such, the click mechanism enabled a relatively large ornithopter of 27.64-gram self-weight with extra 3-gram load to launch a vertical takeoff.

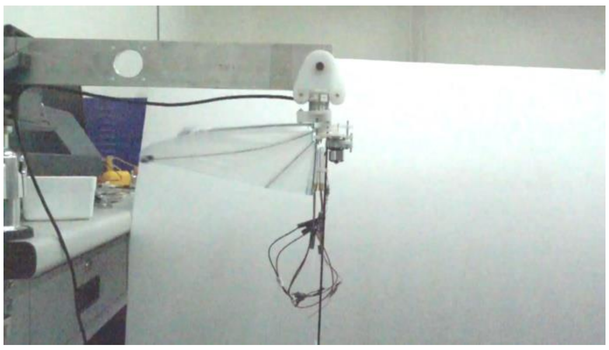


Figure 8: Equipment in force sensor thrust test.

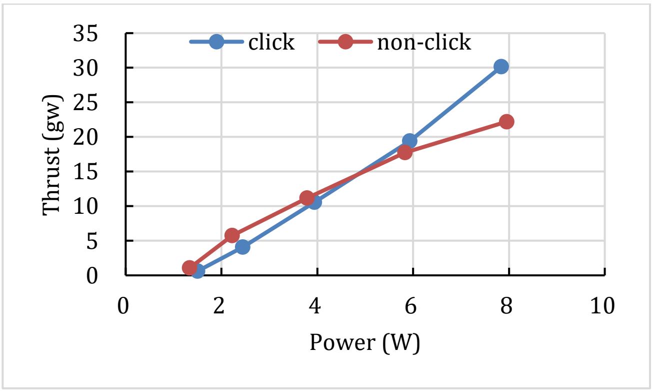


Figure 9: Thrust generated by click/non-click prototype with the corresponding size.

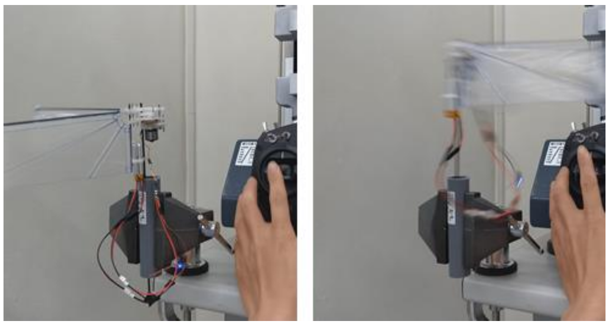


Figure 10: Vertical takeoff in constraint of a PVC tube.

REFERENCES

[1] Chin, Y.W., Kok, J.M., Zhu, Y.Q., Chan, W.L., Chahl, J.S., Khoo, B.C. and Lau, G.K., 2020. Efficient flapping wing drone arrests high speed flight using post-stall soaring. Sci. Robot., 5, p.eaba2386.

[2] Edward G. Boettiger, Edwin Furshpan, "The mechanics of flight movement in diptera" The Biological Bulletin Volume 102, 1952

[3] Thomson, A.J., Thompson, W.A., "Dynamics of a bistable system: The click mechanism in dipteran flight." Acta Biotheor 26, 19–29 (1977)

[4] M.J. Brennan, S.J. Elliott, P. Bonello, J.F.V. Vincent, The ''click'' mechanism in dipteran flight if it exists, then what effect does it have in Journal of Theoretical Biology 224 (2003), 205-213, 2003.

[5] Bin Tang, M.J. Brennan, "On the dynamic behaviour of the "click" mechanism in dipteran flight",Journal of Theoretical Biology, 2011

[6] Chin, Y.W. and Lau, G.K., 2012, October. "Clicking" compliant mechanism for flapping-wing micro aerial vehicle. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 126-131). IEEE.

[7] Yao-Wei Chin and Gih-Keong Lau "Is clicking mechanism good for flapping wing micro aerial vehicle?", Proc. SPIE 8686, Bioinspiration, Biomimetics, and Bioreplication 2013, 86860W (8 April 2013)

Propulsive efficiency of small multirotor propellers in fast forward flight

[1]L. Bullard, [2]S. Watkins, and [3]A. Mohamed

Royal Melbourne Institute of Technology, Melbourne

ABSTRACT

This paper presents a summary of experimental results regarding the propulsive efficiencies of small multirotor propellers, in simulated forward flight conditions. An automated test rig was used in a wind tunnel to measure propeller performance data across a range of flight speeds and angles. Propellers of various pitch were also tested and compared in these conditions. Flight angle was demonstrated to have minimal impact on efficiency within the tested range. Maximal efficiencies were demonstrated at the highest advance ratios and lowest geometric pitches tested.

1 INTRODUCTION

The extremely fast developing small 'sport' UAS industry often produces products with little to no published testing and data. Especially in the bleeding edge developments in small scale sport drones, where efficiency is key, development appears primarily guided by 'feel'. While comprehensive analyses exist for aircraft propellers and even some larger multirotor rotors, very little data is available on the extremely common 5-inch diameter propeller configurations. These configurations typically operate within a low Reynolds number, in highly oblique flow and at a considerably faster velocity than most commercial multirotors. A large volume of propeller variants exists in this regime, with varying blade geometries, blade numbers, pitch and materials. Manufacturers provide little to no information about the performance of these propellers in their target flight regimes, therefore analytic testing can allow for a more educated propeller selection for a given design.

[1] liam.bullard@rmit.edu.au
[2] simon.watkins@rmit.edu.au
[3] abdulghani.mohamed@rmit.edu.au

While multirotors operating within the specified configuration utilizing 5-inch or similar diameter propellers are used primarily by hobbyists, there is a growing demand for smaller scale commercial UAS operations. However, little research has been performed regarding the performance characteristics of small UAS propellers, or even larger propellers in forward flight. Deters, Ananda Krishnan & Selig tested several 5-inch UAS propellers in axial flow, demonstrating a peak efficiency at an advance ratio of approximately 0.6, consistent over the range of tested Reynolds numbers [1], however no consideration was given to propeller performance in oblique flow. Experiments covering oblique flow include those by Theys et al, however testing was performed with larger, 9-inch propeller at low flow velocities [2, 3]. Theys concluded that Blade Element Momentum Theory was impractical, due to the lack of detailed geometry and specifications provided by manufacturers. Amir, Devin & Götz demonstrated varying trends in propeller thrust coefficient with an increase in freestream advance ratio at different flow angles. While these tests were carried out at considerably higher flow velocities, they utilized much larger 18-inch propellers [4]. A variable pitch propeller was utilized by Riccardi in order to minimize variables in propeller geometry, although the rotors used are not representative of the style of propeller commercially available [5].

2 EXPERIMENTAL SETUP

Testing was conducted within the RMIT industrial wind tunnel, using a semi-custom rig to measure propeller parameters in oblique flow.

The test rig selected is based on the commercially available RCbenchmark Series 1580 thrust stand. The thrust stand will produce a log of applied thrust, torque, motor rpm and input power, therefore allowing the efficiency of the motor to be determined. It also allows for custom input functions to be programmed.

The RCbenchmark unit was mounted to the shaft of a large stepper motor, allowing for precise and automated control of the propellers angle of incidence. Propellers chosen are from the HQProp V1S product line, due to their nominally constant blade geometry, over the range of varying pitches available. All chosen propellers have a 5-inch diameter, and 3 blades. 4 propellers were chosen from this series, featuring advertised pitch values of 4, 4.3, 4.8 and 5 inches. This propeller features a product code of the format HQ (diameter)x(pitch), in inches – i.e., HQ 5x4 represents the model featuring a 5-inch diameter and 4 inch advertised geometric pitch. The motor to drive these propellers chosen is the T-Motor F80 2500kv model, as it operates in a relevant RPM range to the chosen propellers, while also offering a larger thermal capacity than motors typically used with this class of propeller. The motor was operated at the nominal voltage of its recommended battery configuration, 14.8V. A T-Motor Flame 80A electronic speed controller (ESC) was used to drive the motor, while the stepper motor was driven with a Geckodrive 6203V driver. Testing was coordinated through the built-in scripting function of the RCBenchmark, allowing for control of both the test motor and stepper motor. This arrangement is depicted in figure 1, with components summarized in table 1.

Force/Torque Measurement [1]	RCBenchmark 1580
ESC [2]	T-Motor Flame 80A
Test Motor [3]	T-Motor F80 2500kv
Power Supply	Chargery Power 1500W (14.8V, 60A)
RPM Sensor	RCBenchmark Back-emf Sensor
Stepper Driver [4] (Stepper motor mounted under thrust stand)	Geckodrive 6203V
Propellers	HQProp 3 Blade V1S Series

Table 1: Testing Equipment

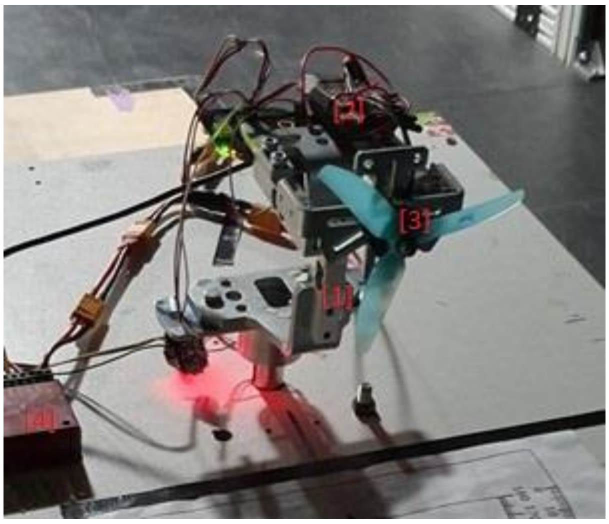


Figure 1: Test rig mounted in wind tunnel.

3 EXPERIMENTAL PROCEDURE

Testing for each propeller occurred at 3 flow speeds, and 6 flow angles, assessed as representative of typical 'sport' multirotor flight regimes, as in Table 2. Hover represents a flow angle of 0°, while traditional fixed wing flight (axial flow) represents a flow angle of 90°, depicted in figure 2.

Wind Speed U	10, 15, 20	[m/s]
Flow Angle	30, 35, 40, 45, 50, 90	[°]
Propeller Pitch	4, 4.3, 4.8, 5	[″]
Rotational Speed	10,000-Max (approx. 30,000)	[rpm]

Table 2: Testing Matrix

Each test was conducted by ramping the propeller through its rpm range, from 10,000rpm to the maximum rpm available, which varied between configurations (approx.. 30,000rpm). This ramp occurred over 45 seconds, and was repeated 3 times. This represented the maximum testing duration due to the thermal constraints of the motor, requiring a cooldown period at the end of each test.

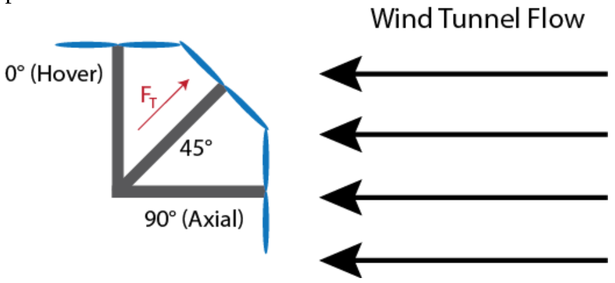


Figure 2: Angle and Force Convention.

4 ANALYSIS

Logged data available from testing is summarized in table 3:

Output Data	Unit
Thrust (F_T)	[N]
Mechanical Torque (M)	[Nm]
Electrical Power (P_{elec})	[W]
Angular Velocity (n)	[Hz]
Temperature (T)	[°K]
Atmospheric Pressure (P_{atm})	[Pa]
Propeller Diameter (D)	[m]

Table 2: Logged Data

$$\rho = \frac{P_{atm}}{RT} \quad (1)$$

Density was first derived using the local atmospheric conditions, for each session of testing. While data was also collected to characterize efficiency of the motor and electronic speed controller, these results were not within the scope of this test. Mechanical power was calculated from the mechanical torque measured by the thrust stand, thus eliminating motor efficiency from further calculations:

$$P_{in} = Mn \; (2)$$

Non dimensional propeller performance coefficients were calculated using the above logged data, derived from those presented by [2, 6]. Thrust and power coefficients are calculated as follows:

$$C_T = \frac{F_T}{\rho n^2 D^4}, C_P = \frac{P_{in}}{\rho n^3 D^5} \quad (3)$$

A final dimensionless coefficient is required, advance ratio:

$$J = \frac{U}{nD} \; (4)$$

Therefore, propulsive efficiency of the propeller can be calculated as:

$$n_{prop} = J\frac{C_T}{C_P} \; (5)$$

These calculations provide a propulsive efficiency value through the full range of RPM values.

4.1 *Error*

In order to determine data quality, a mean and standard deviation was taken for the three tests conducted at each rpm. A sample of this error measurement, for one flow speed and angle is presented in figure 3.

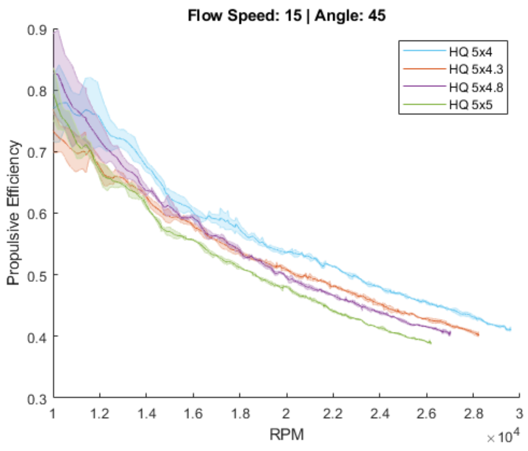


Figure 3: Sample result with standard deviation and mean

As demonstrated, where for each case the solid line represents the mean measurement and the shaded area represents the standard deviation, the highest quality data is present in the region between 18000 and the maximum of 30000 rpm. This is also evident when watching the ramp occur, as the motor reached approximately 20000 rpm with minimal throttle input, resulting in much lower data density in this range. The data was therefore 'cropped' to begin at 15000 rpm to better visualize the high quality data at the higher end of the rpm spectrum, shown in Figure 4

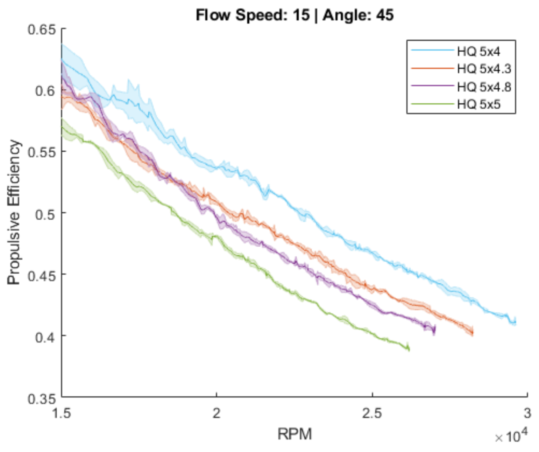


Figure 4: Sample cropped result

5 RESULTS

In all tests carried out, the greatest propulsive efficiency was achieved at the lowest measured RPM. In general, the lowest pitch propellers featured the highest propulsive efficiencies, surpassed in only a handful of tests. These results demonstrate the incompatibility of these propeller designs with efficient flight within their desired operating conditions. While no conclusions can be drawn regarding a clear trend in performance at different angles, this analysis provides evidence that flow angle does not provide a strong advantage to a high or low pitch propeller. Propulsive efficiency, however, is insufficient to fully characterize the performance these propellers. Doing so would require further definitions of flight conditions and requirements, and outside the scope of this work. Data is summarized in Figures 5 through 8.

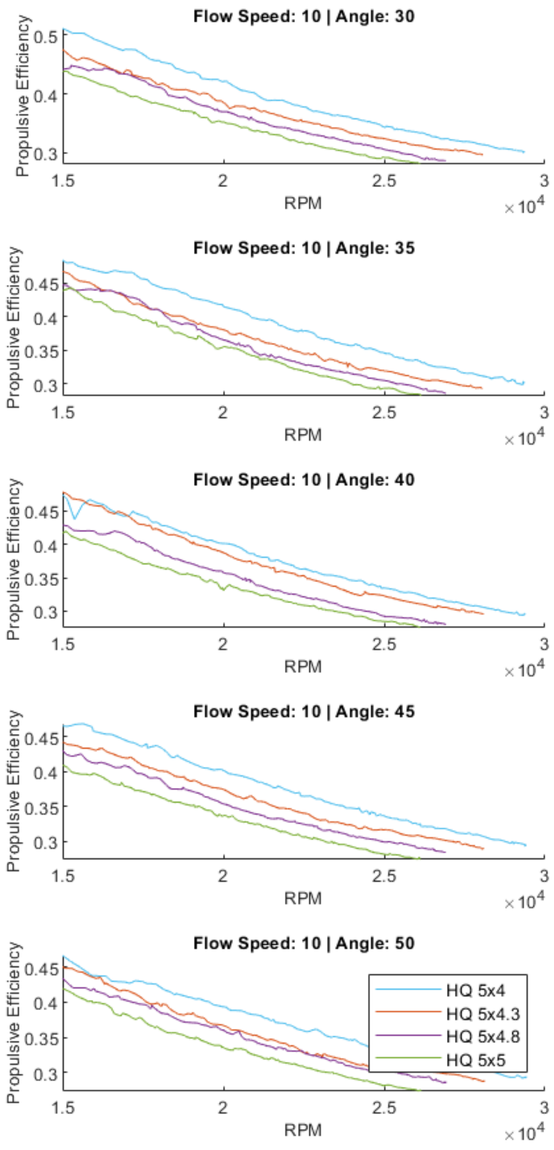


Figure 5: Results at 10 m/s

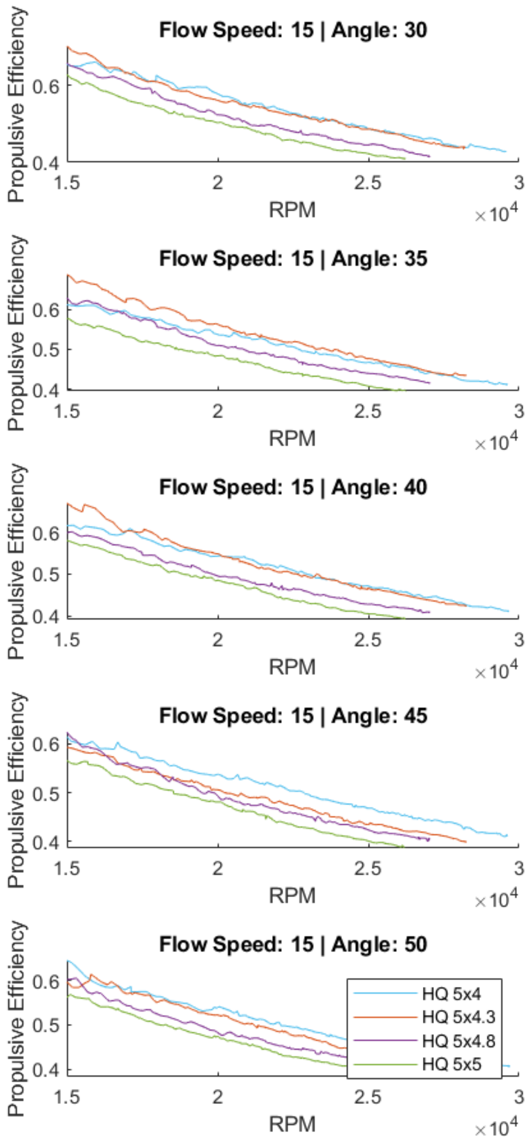


Figure 6: Results at 15 m/s

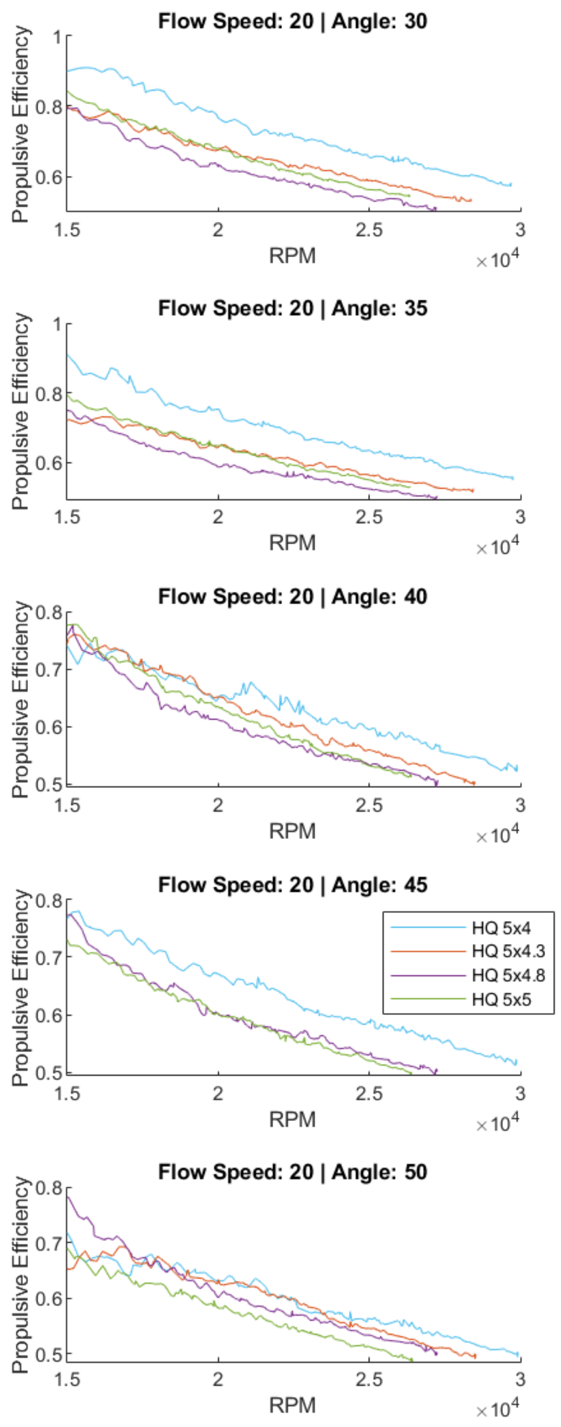


Figure 7: Results at 20 m/s

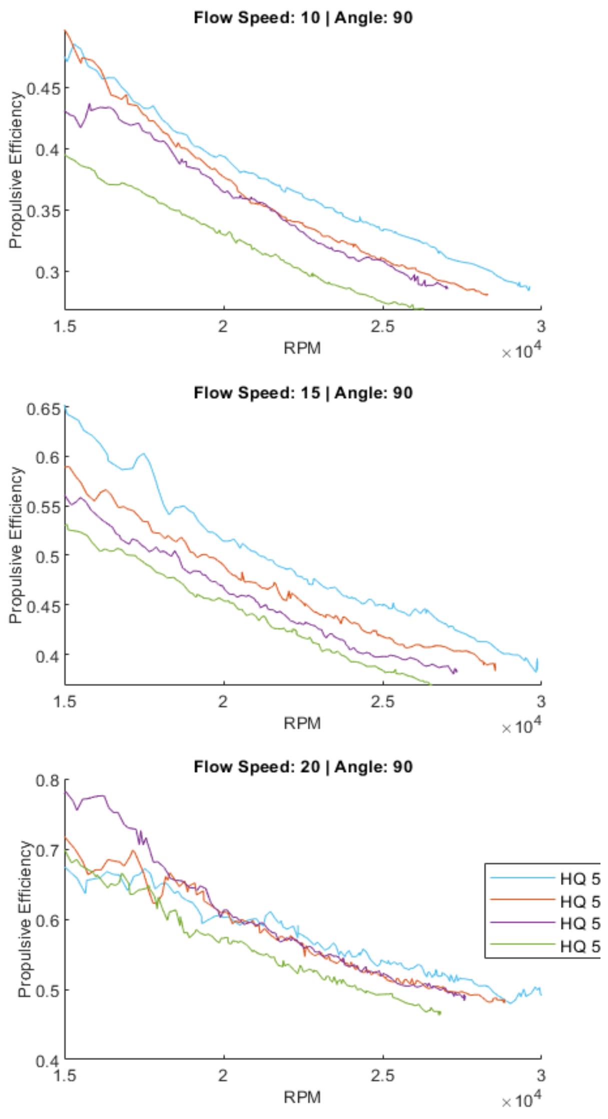


Figure 8: Results in axial flow

[4]5x4.3 Propeller is absent from 20 m/s, 45 degree flow due to erroneous data acquisition

6 CONCLUSION

This paper presents measurements of propeller propulsive efficiency across a range of selected forward flight regimes, such as those experienced by small ‘sport’ multirotors. In the conditions tested, all propellers experienced their maximum efficiency at their highest advance ratios. In a majority of operating modes tested, propellers with lower geometric pitch provided higher propulsive efficiencies. This indicates a greater efficiency could be achieved for this propeller class through a higher forward flight speed, lower pitch, or reduced rpm. This however would require further characterization of propeller performance across thrust values. Additionally, while flight angle within the chosen operating ranges did not have a significant impact on efficiency, the highest efficiencies were achieved in pure axial flow.

Further work in this field including rotor interactions can be viewed in the thesis of Samuel Prudden, available via the RMIT research repository: *Rotor aerodynamic interaction effects for multirotor unmanned aircraft systems in forward flight.*

ACKNOWLEDGEMENTS

The authors thank Samuel Prudden for his valuable advice and feedback during experimental design.

REFERENCES

[1] R. W. Deters, G. K. Ananda Krishnan, and M. S. Selig, "Reynolds number effects on the performance of small-scale propellers," in *32nd AIAA applied aerodynamics conference*, 2014, p. 2151.

[2] B. Theys, G. Dimitriadis, T. Andrianne, P. Hendrick, and J. De Schutter, "Wind Tunnel Testing of a VTOL MAV Propeller in Tilted Operating Mode," *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1064-1072, 2014.

[3] B. Theys, G. Dimitriadis, P. Hendrick, and J. De Schutter, "Experimental and numerical study of micro-aerial-vehicle propeller performance in oblique flow," *Journal of Aircraft*, vol. 54, no. 3, pp. 1076-1084, 2017.

[4] K. Amir, B. Devin, and B. Götz, "Experimental Analysis of a Small-Scale Rotor at Various Inflow Angles," *International Journal of Aerospace Engineering*, vol. 2018, 2018.

[5] F. Riccardi, "Wind Tunnel Testing of a Variable-Pitch Quadrotor UAV Isolated Rotor", in *41st European Rotorcraft Forum*, Munich, Germany, 2015, p. 14.

[6] J. Leishman, “Principles of Helicopter Aerodynamics,” New York, NY: Cambridge University Press, 2000.

Aero-propulsive performance improvement of H_2 powered UAS

Nikola Gavrilovic[*], Phassawat Leelaburanathanakul, Javier Cuadrado, Jean-Marc Moschetta
ISAE-SUPAERO, University of Toulouse, 31400 Toulouse, France

ABSTRACT

Performance analysis and possible improvements of a long-range unmanned aircraft system powered by fuel cell are investigated using CFD, with the study focusing on the feasibility of crossing the Atlantic Ocean. The motivation behind this aircraft is to demonstrate the capability of the hydrogen fuel cell as an alternative fuel source and to create a case for future commercial and civilian aircraft. The existing hydrogen-powered UAS design is benchmarked and an in-depth analysis of several aerodynamic structures for performance improvement in cruise. The requirements of a 3000 km range, a maximum mass of 25 kg, and hydrogen as a primary energy source, are used as inputs for the conceptual design phase and performance evaluation. The propulsion set, including the propeller geometry and the electric motor, has been optimized for cruise conditions. A detailed study of integrated propeller emplacement has been investigated, showing significant benefits in efficiency.

1 INTRODUCTION

The project Drone Mermoz aims to analyze the feasibility of an unmanned aircraft system powered by hydrogen fuel cells that have the capability of crossing the Atlantic Ocean. This route has been selected as it has historical significance; it was used by the French aviation company, Aeropostale in the 1930s and to date has only been crossed by UAS powered with internal combustion engines. The objectives of this project are to design a long-range UAS featuring hydrogen fuel cell-based propulsion, capable of flying from Dakar to Natal (3000 km) and being sufficiently lightweight to be within the certification category allowing beyond the line of visual sight.

Unmanned aircraft systems (UAS) have become instrumental tools for missions in various military, civil and commercial fields. Current generation electrical powered unmanned aircraft systems are limited in terms of range and endurance due to the low energy density of their lithium-based batteries. However, many UAS applications require high range and endurance capabilities for intelligence, surveillance and reconnaissance. This demand for flights which last for considerable periods of time without the need to frequently land coupled with efforts to minimize environmental impact and the benefits of a low thermal and noise signature, make long range electrical aircraft desirable. An emerging source of electrical energy with the potential to solve the limitations of batteries is hydrogen fuel cells. They offer compelling value for unmanned aircraft systems due to the ability to provide approximately five times more power per flight hour for the same weight as lithium based batteries, as well as offering improved reliability and reduced maintenance when compared to small internal combustion engines. Some recent example of hydrogen powered aircraft can be found in [1, 2, 3, 4, 5, 6, 7].

[*]Email address(es): nikola.gavrilovic@isae-supaero.fr

2 CONTEXT

2.1 *Performance improvement of a clean aircraft*

A preliminary design study of an ultra-long-range drone capable to cross the Atlantic ocean by using fuel cells and hydrogen as a primary energy source has been investigated previously by Gavrilovic et. al. [8] and is shown in Figure 1.

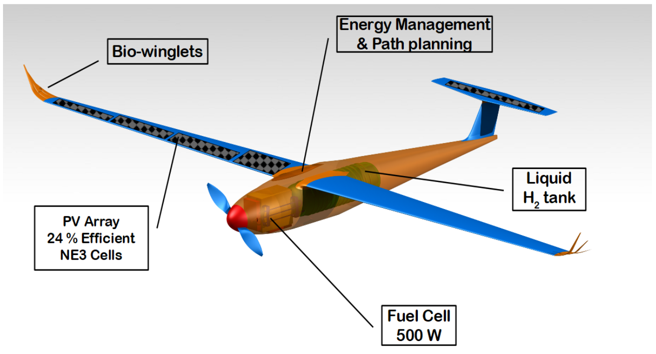

Bio-winglets
Energy Management & Path planning
Liquid H_2 tank
PV Array 24 % Efficient NE3 Cells
Fuel Cell 500 W

Figure 1: Drone Mermoz v1 - 12 kg and 3.6 m span demonstrator of technology.

Further development of a design procedure led to a combination of analytic approach and optimization cycle. The structural mass estimation was performed using the Gundlach [1] equations, with the iteration of the propulsive system in the function of available energy required for the journey. Once the estimated mass was determined, a parametric study was conducted to find adequate ranges of aircraft size which have been used in the optimization cycle. The last part of the preliminary design procedure was an optimization cycle working with a modified version of AVL which takes into account viscous effects integrated into an OpenMDAO genetic algorithm environment.

A final result of the optimization cycle was a 12 kg aircraft with wing-span of 3.6 m, having maximum lift-to-drag ratio of around 25. The clean aircraft named Drone Mermoz v1 and shown in Figure 2 is designed for following mission requirements:

- Must be able to cross the distance of 3000 km with liquid hydrogen as a primary energy source.
- To have a total mass of less than 25 kg.
- Must use hydrogen fuel cell as primary energy source

The performance improvement of this given clean aircraft is related to design, development and integration of certain aerodynamic structures. The main objective is to improve the aircraft endurance and range. Therefore, the CFD analysis of this clean aircraft will be performed in order to be compared with improved design such as:

- Fuselage-wing junction optimization (design of "Karmans")
- Implementation of winglets (comparison between bioinspired wing-tip feathers and blended winglet).

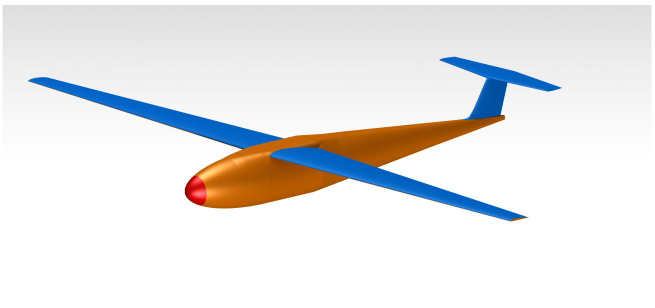


Figure 2: Drone Mermoz v1 - clean aircraft.

The main geometrical, aerodynamic and propulsive parameters of Drone Mermoz v1 are presented in table 1.

2.2 *Propulsive optimization*

The main objectives of the propulsive optimization study are a choice of the propeller, motor, and a more interesting part of fuselage integration that will be presented here. This study aims to investigate the the effect of fuselage body on the performance of the propeller according to different emplacements. A potential benefit in increased propulsive efficiency and thrust can come from two different points. The first is related to a covering of around 25 % of propeller root as this part mainly generates drag, while the second is related to the investigation of flow deviation to overall propeller performance. Therefore, the study will focus on propeller integration in already existing fuselage design (without fuselage shape modification), thus avoiding standardized spinners.

Parameter	Value	Unit
V_c	23	$[m/s]$
Wingspan	3.6	$[m]$
Fuselage length	1.7	$[m]$
Wing surface	0.65	$[m^2]$
AR	20	—
Total mass	12	$[kg]$
Structural mass	4.5	$[kg]$
$C_{Lcruise}$	0.57	—
$C_L/C_{D_{max}}$	25	—
ρ	1.2	$[kg/m^3]$
Fuel cell	HES AEROSTAK 500	—
LH_2 reservoir	7	$[l]$
Range	3200	$[km]$

Table 1: Drone Mermoz V1.

3 Geometry and Mesh Preparation

3.1 *Boundary conditions and domain*

The dimensions of the domain are chosen to be sufficiently large to allow enough space around the geometry of interest so that the perturbations in the flow field do not interfere with the boundaries. The domain shown in Figure 3 was selected so that the domain should allow a minimum of 2 times the geometry's length in the upstream direction, 5 times in the downstream direction, and 2 times in width.

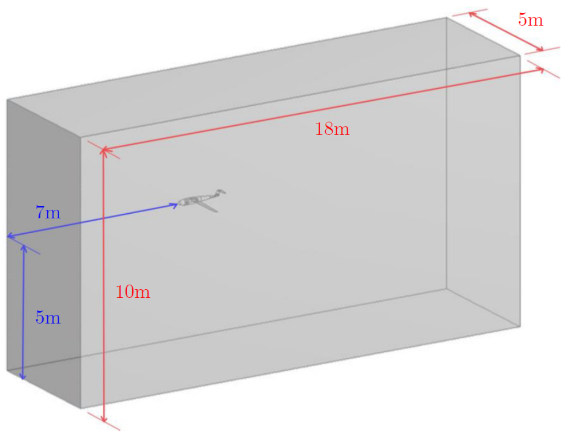


Figure 3: Flow domain.

After the definition of the physics continua to be used for the computation, the boundary conditions of the mesh domain are specified. Using the surface parts that were named during the mesh construction process in the ICEM-CFD software, the definition of types, physics conditions, and values of the boundary condition is straightforwardly inputted into the solver software. A diagram describing the different boundary conditions used for simulations with a positive angle of attacks is shown in Figure 4 below.

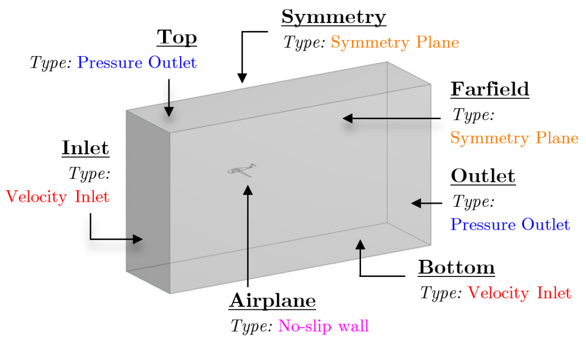


Figure 4: Boundary conditions.

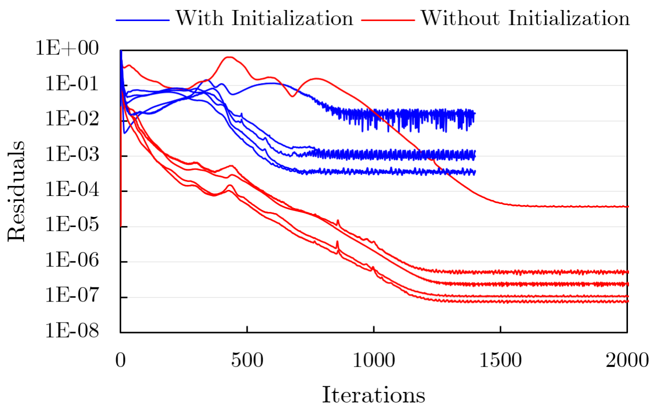


Figure 5: Residuals.

3.2 *Computational setup*

For the simulations without an angle of attack, the Top, Bottom, Symmetry, and Farfield boundary condition type is set to a symmetry plane. In STAR-CCM+, the tangential shear stress at a symmetry boundary is fixed to zero. The velocity face value at the boundary is extrapolated from the parallel component of the flow of the adjacent cells using reconstruction gradients. This ensures that there will be no flow passing through a symmetry boundary. The methodology of the pressure outlet boundary condition is similar as it also extrapolates the velocity of the interior cells to the boundary face using reconstruction gradients. Since the reference pressure of the simulation is already fixed at 101,325 Pa, the Outlet pressure specification is set to 0 Pa gauge.

The criterion that used to ensure the solution's convergence is the residuals shown in Figure 5 of the transport equations. In CFD analysis, residuals quantify the local imbalances of variables at each control volume. The velocity inlet flow direction was specified using normalized x and y components. For the simulations with a negative angle of attack on the airplane, the Bottom boundary condition is switched to a pressure outlet and the Top boundary condition is set to a velocity inlet. For an airplane model, the angle of attack range is from $-4°$ to $+12°$, giving a total of 12 simulations. Moreover, an extra simulation will be conducted at the drone's operating point.

3.3 *Mesh study*

This chapter will provide an extra effort in finding out the balance between the solution accuracy and the computational time. It must also be pointed out that the time required to mesh geometry is not insignificant either. For the mesh size used in this project shown in Figure 6, a full mesh generation time on a personal computer takes approximately 1.5 to 2 hours, including the volume and prism generation. This value can extend up to 2.5 hours when generating the finer mesh for the convergence study. The geometry used for this project's mesh independence study is the aircraft shown in Figure 2. The mesh is varied by re-sizing the parts mesh setup, as well as the volumetric refinement on all density regions. There is a total of 5 mesh variations, gradually increasing from 8.5 million cells to 15.9 million cells. Since the prism layers are

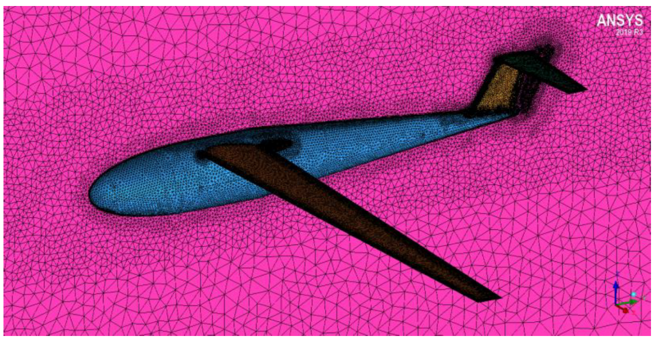


Figure 6: Surface mesh on the first baseline (v1) version of the airplane.

sized according to the flat-plate boundary layer theory, it is crucial that the wall y+ is verified after the simulation is computed to see if further refinements are necessary. To ensure that there is at least one cell to resolve the flow within the viscous sub-layer of the boundary layer, the wall y+ should not exceed 5. Figure 7 shows the distribution of the wall y+ of all cells adjacent to the airplane's surface. It can be observed that a very large quantity of the cells on the airplane's surface has a y+ value of around 1. As a matter of fact, 77.0 % of all adjacent cells has a y+ value less than 1.2 and 99.0 % has a y+ value less than 1.6. There are only 2 from the total 198,827 adjacent cells that has a y+ value greater than 5 which can be safely considered negligible.

Additionally, a closer inspection of the wall y+ value of the cells on the fuselage at the symmetry plane in Figure 8 shows that most of the cells are less than or equal to 1. The dimensionless velocity profile of the flow above the fuselage at X-position equal to 0.637 meters shown as a magenta line in Figure 8 is examined.

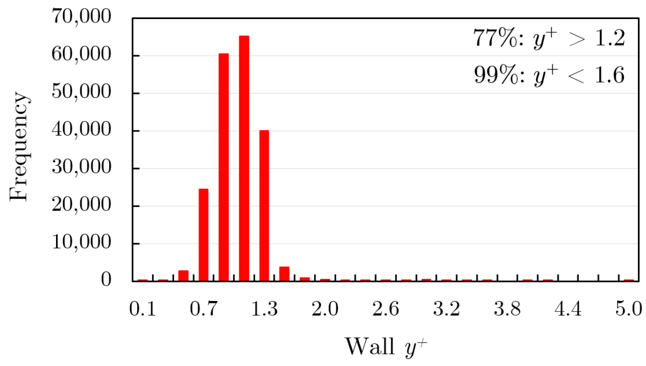


Figure 7: The wall y+ distribution of airplane-adjacent cells.

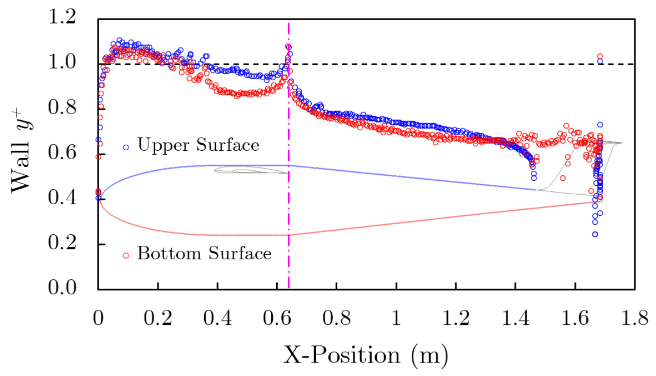


Figure 8: The wall y+ value of the fuselage top and bottom surface at the symmetry plane.

4 Results

4.1 General performance

An initial study has been performed to analyze and compare the general performance of the aircraft using both vortex lattice program AVL and previously explained CFD setup. The lift slope curve shown in Figure 9 shows a slight difference in zero angle of attack lift coefficient while having the same lift slope.

The main difference between two methods is shown in Figures 10 and 11 and is coming from difference in drag prediction. A vortex lattice program used in this work is a modified version of AVL, which includes the prediction of viscous drag, where the viscous drag coefficient $c_{vd} = c_{vd}(Re, \alpha_t)$ depended on a chord-based Reynolds number and the total angle of attack α_t. The fact that the total drag prediction of modified AVL depends on the airfoil viscous database previously built, and a choice of default profile drag coefficient added to geometry brings a certain doubt in total coefficient values. On the other hand, with a calculation time of less than a second, the potential for comparative studies and ease of integration in the optimization loop keep modified AVL as a highly desirable tool, especially in the preliminary design phase.

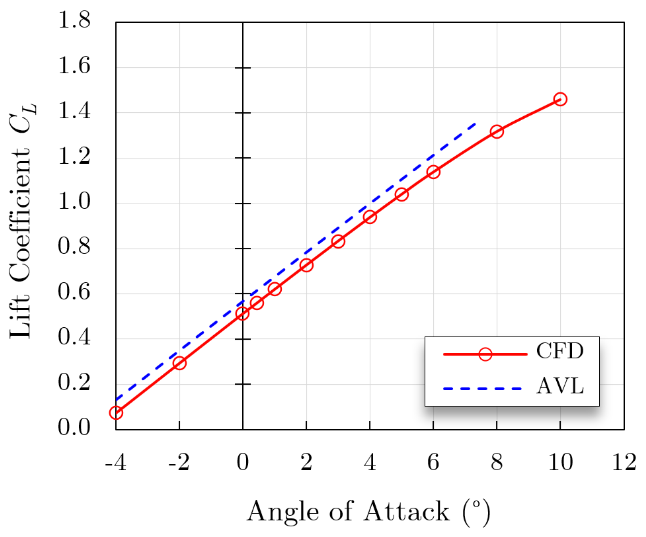


Figure 9: Lift curve.

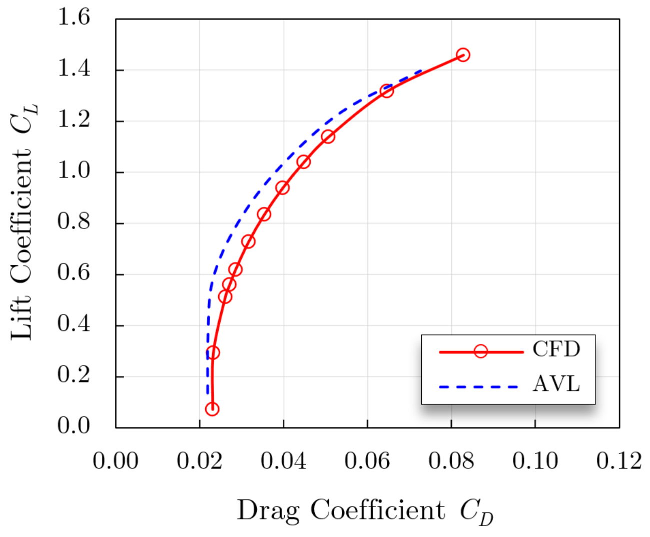


Figure 10: Polar.

The Figure 12 shows a span-wise lift distribution for a trapezoidal wing shown in 2 which is almost elliptical for a chosen tapper ration of 0.36.

Further analysis of fuselage pressure coefficient distribution shown in Figure 13 revealed a small contribution in a lift in the area ahead of the wing. The plot also reveals a peak of a pressure coefficient at the place after wing-fuselage junction where the transition to a rear part of the fuselage-cone is beginning. The conclusion is that this kind of sharp geometrical transition should be avoided in the definitive version which will be fabricated.

Moreover, a total drag decomposition shown in Figure 14

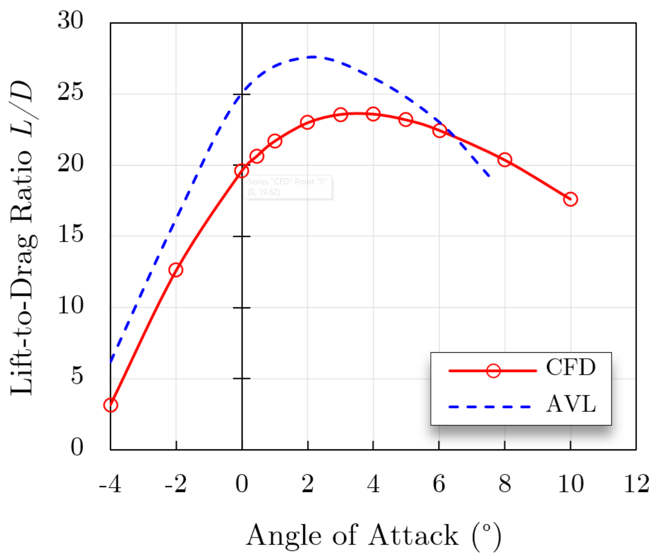


Figure 11: Lift-to-drag ratio.

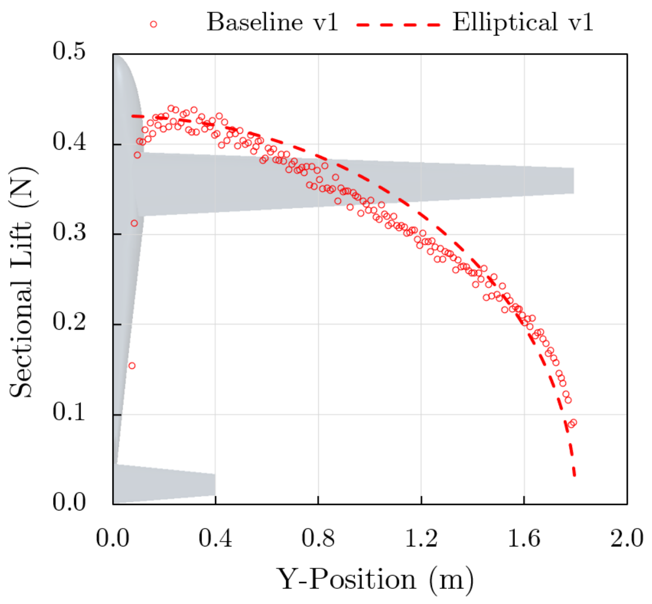


Figure 12: Span-wise lift distribution.

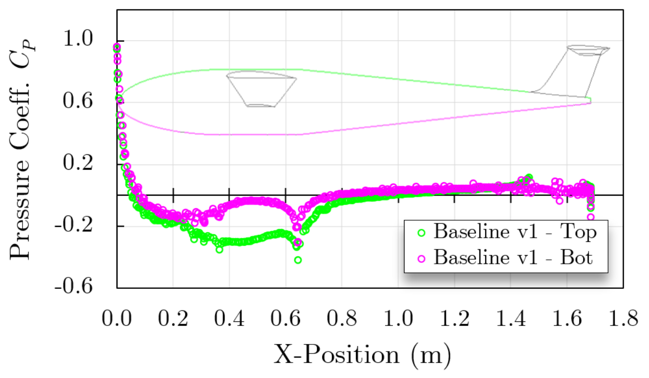


Figure 13: Fuselage pressure distribution.

shows that a 60 % of a total drag is coming from a viscous part, while the other 40 % belongs to a pressure drag. Further analysis of viscous drag decomposition shown in Figure 15 revealed that the majority of viscous drag is coming from a wing due to the biggest part of its wetted surface when compared to other parts of the aircraft.

Finally, a pitching moment coefficient has been shown in Figure 16 for various reference locations. The objective of this study was to determine where is the position of the neutral point and to compare it with prediction coming from AVL. As it can be seen, a point for which the pitching moment coefficient does not vary with angle of attack is located at $x = 0.510m$ from an aircraft nose. This value is less than a 3 % difference than the one coming from an AVL, which moreover confirms the benefits of using AVL in the preliminary design phase.

4.2 Improvement using winglets and karmans

This chapter has been devoted to the exploration of potential aerodynamic structures that could enhance the overall performance of the aircraft, therefore, endurance and range. The previous study shown by Gavrilovic [9] has quantified considerable improvements that can be achieved using winglets on commercial aircraft. However, due to a huge discrepancy in flight conditions, a new study has been conducted adapted to flight conditions of a small drone. It should be noted here that even a benefit of a couple of percent is highly valuable as the aircraft is supposed to fulfill the mission requirement of having more than 3000 km of range. Two different winglet designs have been studied and compared to clean aircraft performance. The first one is a bio-inspired winglet shown in Figure 17 that resembles the eagle wing tip feathers. The second design is a classical blended winglet shown in Figure 18, found on various aircraft types, from small UAVs for lateral stability purposes up to big commercial aircraft for induced drag reduction. Moreover, a smooth karman design have been presented in Figure 19. On top and below the wing it consists of small rounded edge to reduce the surface and such friction drag. At the leading and trailing edge it consists of much larger taper and smooths out the pressure differences: High pressure at the leading and trailing edge, low pressure on top of the wing and around the fuselage. The main objective of the karman is to suppress potential formation of vortex in the rear part of the fuselage-wing junction, and therefore, prevent possible drag sources.

The final comparison between different structures and clean aircraft for cruise operating conditions is presented in Figure 20. It can be seen that the karman design provided around 1 % reduction in total drag. On the other side, winglet structures provide more significant benefits with up to 5 % in

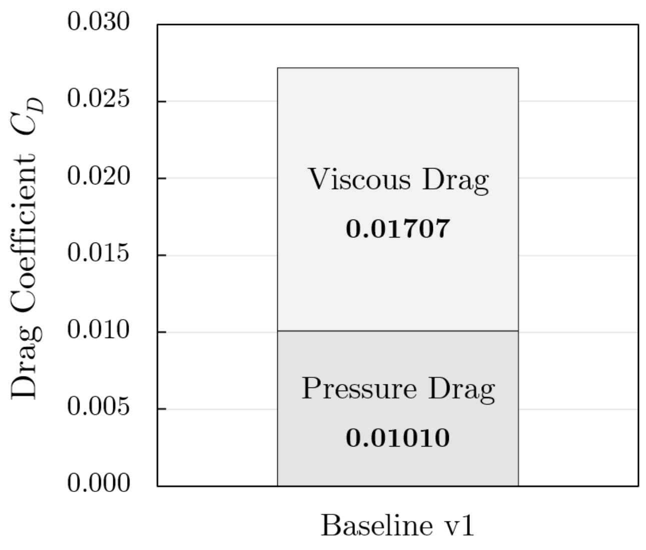


Figure 14: Total drag of the aircraft in cruise.

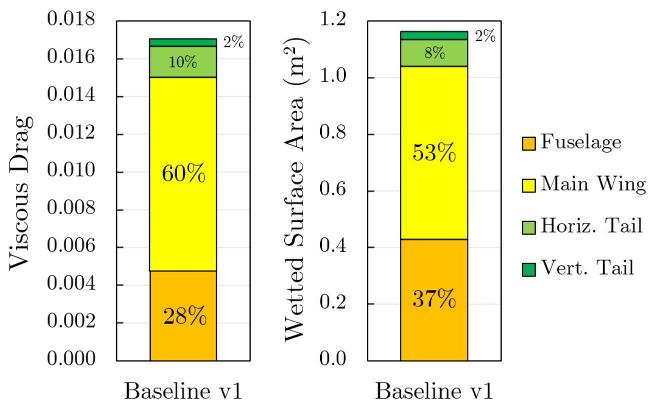


Figure 15: Viscous drag of the aircraft in cruise.

total drag reduction. Moreover, the addition of winglets usually brings around 4-6 % higher lift coefficient for the same angle of attack. That means that the drag can be even more reduced by decreasing the required angle of attack in cruise for a configuration with winglets. Finally, it should be pointed out that a 5 % drag reduction represents a significant achievement, being 150 km out total required 3000 it can be considered as a fuel reserve.

4.3 Propulsive optimization

Early design stages focused on finding the optimal propulsive system (propeller + motor combination) for the desired task. Indeed, given the very long-range to achieve, even the slightest improvements of performances can dramatically affect the final output of the mission. To this end, all the propellers listed on the *APC Propeller* website were tested, alone and in combination with the electric *T-Motors* and *Aximotors*, to find the best possible propulsive system.

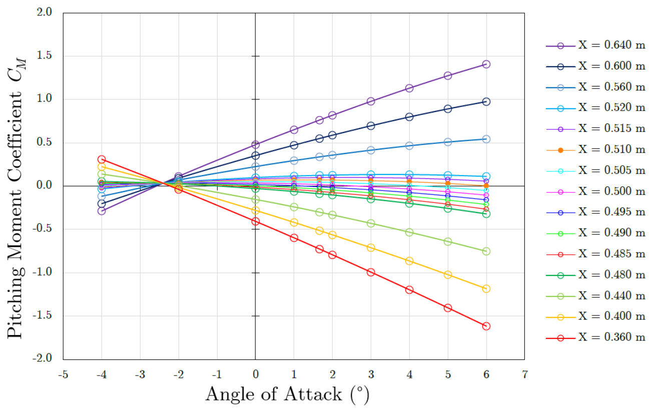


Figure 16: Pitching Moment Coefficient CM vs. angle of attack measured at various points along fuselage axis.

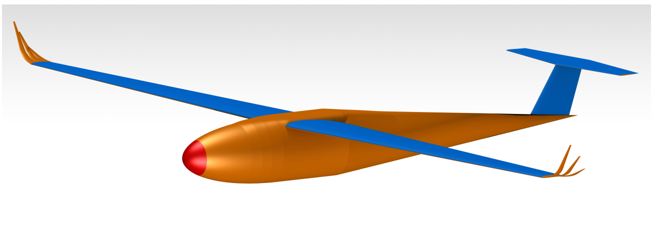


Figure 17: Bio-winglet.

Optimal propellers were found to be around the 20-inch diameter range when rotating at around 3000 rpm. Smaller blades require greater rotational speeds for similar performances, while greater ones can lead to mass increments detrimental to the desired range. While our interest remains still to design an optimal propeller for the desired task, this analysis allows us to identify a promising range of propeller dimensions to analyze, thus making the final design process easier.

Finally, we found that the availability of commercial electric motors is sufficient to find a propeller/motor combination with sufficient performances for the desired task. Furthermore, in case of need, the construction of a dedicated motor is feasible, so we decided to proceed with the design of the optimal propeller, as well as an analysis of the influence of its position in the fuselage in its aerodynamic performances.

We now proceed to study the aerodynamic performance

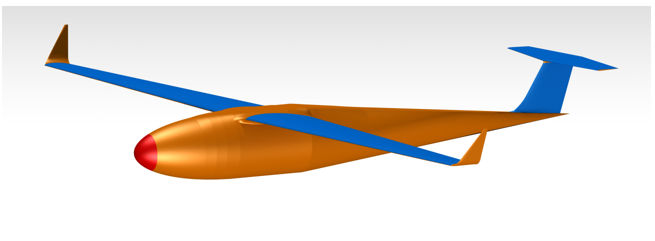


Figure 18: Blended-winglet.

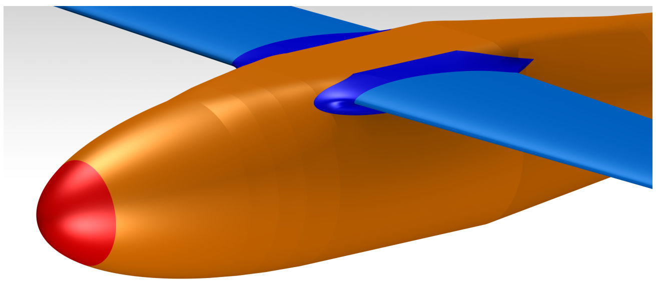


Figure 19: Wing-fuselage junction.

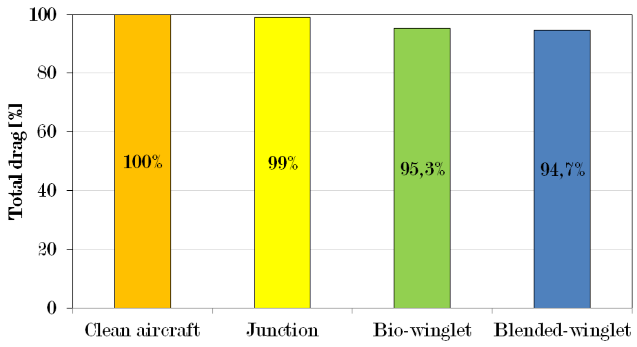


Figure 20: Total drag for different aircraft configurations.

of the optimum propeller, obtained for Drone Mermoz v1, when embedded 50 mm away from the tip of the fuselage. Having verified the viability of using XRotor with a cho-

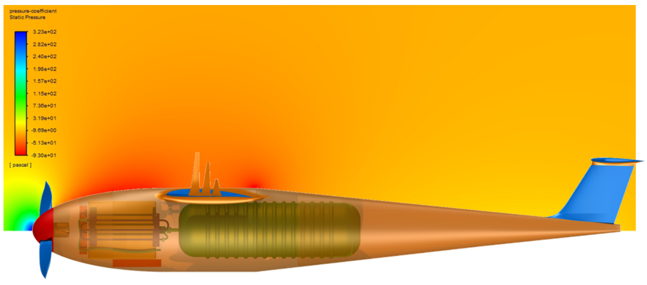


Figure 21: Fuselage pressure distribution.

sen propeller, we proceed to analyze the aerodynamic performance of chosen propeller when embedded in a spinner 50 mm away from the aircraft's nose. XRotor can compute the behavior of a propeller when under a non-uniform stream by being given the flight speed and the additional speed for each radial station. To obtain this speed distribution at the station x=50 mm, an Ansys-Fluent software package has been used. After analyzing the propeller for the non-uniform stream, we obtain the following outcome:

Having seen the beneficial effects of embedding the propeller in a spinner, we seek now to determine the optimal

Parameter	Isolated	Embedded	increment %
Thrust [N]	4.67	4.85	+3.85 %
η_p **[%]**	87.49	91.62	+4.13 %

Table 2: Performance variation of a chosen propeller.

position to place the propeller to fully optimize the performance. To this end, an Ansys-Fluent analysis have been carried out. When moving the propeller backwards, we see that the propeller performances fast degrade. Only in the rearrest part of the ellipsoidal section of the fuselage do we see a slight upgrade of performances when compared with the immediate preceding ones, even if these performances are still worse than those obtained for a propeller embedded 50mm away from the nose, or even an isolated propeller under a uniform stream. The opposite situation is seen when moving the propeller towards the nose, since we obtain an upgrade on the performances. However, there has to be an optimum for these performances, after which they will decay to the values obtained for the uniform stream performances. The evolution of the performances can be seen in Figure 22 and 22.

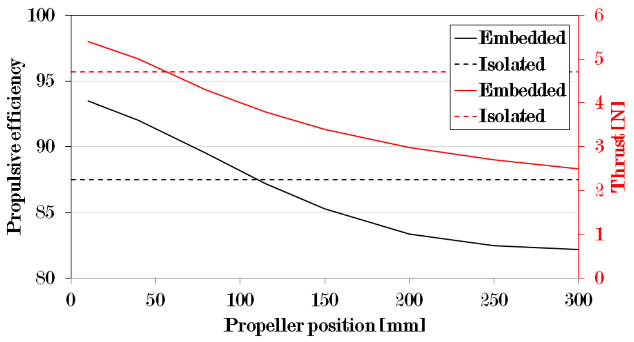


Figure 22: Propeller efficiency and thrust variation with position.

Some conclusions can be extracted from these results:

- While moving the propeller close to the nose yields higher thrusts and efficiency, the increase of thrust also means an increase in the necessary power to fly at the desired speed and omega.
- The most interesting region to place the propeller is the interval $46 < x < 60$ mm, since in this region, the thrust obtained is higher than the one provided by the isolated propeller, while the necessary power to produce this thrust is lower.

Figure 23 show that the propulsive efficiency decreases as we embed a higher section of the propeller into the spinner. This tendency is actually counter-intuitive, since suppressing the inner sections of the blade (which mainly generate drag) should actually lead to increased propulsive efficiency. We, therefore, conclude that the performance variations over the isolated solution are due to the non-uniform velocity profile rather than to suppressing the blade's inner part.

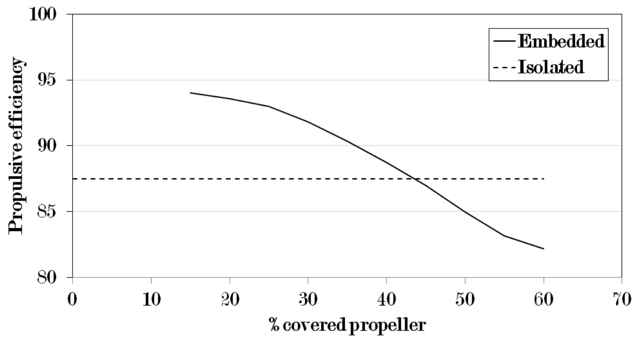


Figure 23: Propulsive efficiency vs. percentage of covered blade for the embedded propeller.

5 Conclusion

A performance study of an ultra-long-range drone powered by a fuel cell and hydrogen has been performed using both CFD and vortex lattice methods. The main findings of this study are that significant achievements in drag reduction can be achieved using both blended and bio-inspired winglets. A drag reduction of around 5 % represents a significant gain in overall performance as it can be taken as a fuel reserve in the mission. On the other hand, fuselage-wing junction design brought benefit in a drag reduction of only 1 % which is still significant and to be confirmed in the wind tunnel campaign. A reasonable match was found in neutral point estimation between the two methods with the conclusion that AVL can be further used for such estimations with confidence. Moreover, a study of integrated propeller design showed that the elliptical fuselage tip can contribute to around 4 % gain in propulsive efficiency of a propeller. The benefit of the increased efficiency and thrust of the propeller was found to be due to coverage of around 25 % of the propeller root and deviated flow due to fuselage presence. The next phase of performance investigation will be a wind tunnel campaign of full-scale aircraft, with objective to confirm and verify gains due to application of new aerodynamic structures and propulsion integration.

References

[1] J. Gundlach. *Designing Unmanned Aircraft Systems: A Comprehensive Approach*. American Institute of Aeronautics and Astronautics, Reston, VA, 2014.

[2] P. Osenar, J. Sisco, and C. Reid. Advanced propulsion for small unmanned aerial vehicles. Technical report, Ballard Canada, 2017.

[3] Sparkle Tech. Eagle plus vtol 3.5m, 2019. http://www.sparkletech.hk/electric-power/eagle-plus-vtol/.

[4] K. Swider-Lyons, K. MacKrell, J. A. Rodgers, G. S. Page, M. Schuette, and R. O. Stroman. Hydrogen fuel cell propulsion for long endurance small uavs. In *AIAA Centennial of Naval Aviation Forum.*, 2011.

[5] H3 Dynamics. Hywings, 2016. http://www.h3dynamics.com/products/hywings/.

[6] Insitu. Scan eagle capabilities, 2018. https://www.insitu.com.

[7] M. Gatti. Complete preliminary design methodology for electric multirotor. *Journal of Aerospace Engineering*, 30(5):1–15, 2017.

[8] N. Gavrilovic, D. Vincekovic, and J-M. Moschetta. A long range fuel cell/soaring uav system for crossing the atlantic ocean. In *IMAV 2019*, 2019.

[9] N. Gavrilovic, B. Rasuo, G. Dulikravich, and V. Parezanovic. Commercial aircraft performance improvement using winglets. *FME Transactions*, 43(1):1–8, 2015.

Implementation of copter propeller model to the problem of energy consumption minimization during lift phase

S.V. Serokhvostov[1]* and T.E. Churkina[2]
[1] Moscow Institute of Physics and Technology (National Research University),
9 Institutskiy per., Dolgoprudny, Moscow Region, Russia
[2] Moscow Aviation Institute (National Research University),
4 Volokolamskoe shosse , Moscow, Russia

ABSTRACT

The problem of minimum energy consumption for the lift phase of copter or VTOL airplane flight is investigated. Two models of propeller for the analytical investigation are proposed. Problem was solved analytically, the results analysis was made and propeller models accuracy was investigated.

1 INTRODUCTION

One of the tasks for copters and VTOL airplanes is to lift some payload to the defined altitude. This task can be a part of the flight mission or the main purpose of the flight. The best way of such a lift process is the flight with minimal energy consumption as it enables to use the accumulators with less mass and/or save energy for other parts of flight mission.

2 PROPELLER MODEL

Propellers used in copters are designed for long-time hovering. They have rather low Pitch/Diameter ratio (about 1:2 or less) that leads to the absence of flow separation on the blades at least during the hovering phase.

To describe the behavior of propellers, dimensionless characteristics are used [1]:
thrust coefficient

$$C_T = \frac{T}{\rho n^2 D^4}, \tag{1}$$

power coefficient

$$C_P = \frac{P}{\rho n^3 D^5}, \tag{2}$$

advanced ratio

$$J = \frac{V}{nD}, \tag{3}$$

propeller efficiency

$$\eta_{prop} = \frac{C_T J}{C_P}, \tag{4}$$

where T — propeller thrust, P — propeller power, ρ — air dencity, n — propeller frequency of rotation, D — propeller diameter, V — air velocity at infinity.

*Email address: serokhvostov@phystech.edu

The analysis of dimensionless characteristics of such a propellers shows that the thrust coefficient C_T decreases practically linearly with advanced ratio J up to zero value (it was shown in [2]) and power coefficient remains practically constant for zero and small values of J. To illustrate this, the plots for some propellers are presented in Figures 2–8 (see Appendix A). Figures 2– 6 show the caracteristics of different propellers with the same pitch and diameter. Figures 6 – 8 show the caracteristics of propellers of the same type, the same diameter and the same manufacturer with different pitch. More examples one can find in [1].

So, the following model is proposed: thrust coefficient decreases linearly with J, power coefficient is constant. In this case only three values are required for the propeller description:

- C_{P0} — power coefficient,
- C_{T0} — thrust coefficient at zero J,
- J_0 — value of J for zero thrust coefficient.

In this notation formulas for C_T and C_P can be expressed as

$$C_P = C_{P0} = \text{const}, \tag{5}$$

$$C_T = C_{T0}\left(1 - \frac{J}{J_0}\right). \tag{6}$$

According to (5), (6), if the copter is hovering or moving vertically with constant velocity V, power P and thrust of propeller T can be expressed as

$$P = kn^3, \tag{7}$$

$$T = an(n - bV), \tag{8}$$

where

$$k = C_{P0}\rho D^5, \tag{9}$$

$$a = C_{T0}\rho D^4, \tag{10}$$

$$b = (J_0 D)^{-1}. \tag{11}$$

From (8) one can find the rotational frequency of copter propeller as a function of T and V as

$$n = \frac{bV + \sqrt{(bV)^2 + \frac{4T}{a}}}{2}. \tag{12}$$

During the hovering or vertical lift with constant velocity the thrust must be equal to

$$T = \frac{mg}{N}, \tag{13}$$

where N is the number of propellers in copter, m is a copter mass, g — gravity acceleration.

3 Energy minimization

Assume that copter must lift to the altitude h with constant velocity (excluding the beginning and the end of the lift), so lift time t is

$$t = \frac{h}{V}. \tag{14}$$

Total energy consumption is

$$E = \frac{NPt}{\eta}, \tag{15}$$

where η is the efficiency of electrical part of powerplant. Assume that η is constant. Substituting (12), (14) into (15),

$$E = \frac{kt}{8\eta}\left(\frac{bh}{t} + \sqrt{\left(\frac{bh}{t}\right)^2 + \frac{4T}{a}}\right)^3. \tag{16}$$

The condition of minimum of energy in this process is

$$\frac{dE}{dt} = 0. \tag{17}$$

Substituting of (13), (16) into (17) gives

$$t_m = bh\sqrt{\frac{2a}{T}} = bh\sqrt{\frac{2aN}{mg}}. \tag{18}$$

Condition (18) together with (12) gives the frequency of optimal lift n_m as

$$n_m = \sqrt{\frac{2T}{a}} = \sqrt{\frac{2mg}{aN}}. \tag{19}$$

It should be noted that this value of frequency is $\sqrt{2}$ times higher than the frequency for hovering.

Propeller power for optimal lift is

$$P_m = k\left(\frac{2T}{a}\right)^{3/2} \cdot N = \frac{k(2mg)^{3/2}}{\sqrt{N}}. \tag{20}$$

It is $\sqrt{8}$ times higher than the propeller power required for hovering.

Formulas (19), (20) give the method of defining the rotation frequency and power in climb. One can measure these values at hovering and simply multiply on the corresponding coefficients.

Also (19), (20) show that for the fixed diameter and geometry of propeller, the more propellers are used (in other words, the more the total area of all the propellers) the lower is the total power and the higher is the time of climb.

One more thing should be mentioned. Condition (18) correspondes to the maximal thrust at fixed V in expression (8).

The value of minimal energy E_m required for lift is

$$E_m = \frac{4kbThN}{a\eta} = \frac{4kbmgh}{a\eta}. \tag{21}$$

As mgh is usefull work in our case, $a\eta/(4kb)$ can be assumed as total effitiency of this process. Using (9)–(11) one can express (21) in the form of

$$E_m = \frac{4ThN}{\eta}\frac{C_{P0}}{C_{T0}J_0} = \frac{4mgh}{\eta}\frac{C_{P0}}{C_{T0}J_0}. \tag{22}$$

So the other definition of the process total eficiency η_{total} is

$$\eta_{total} = \eta\frac{C_{T0}J_0}{4C_{P0}}. \tag{23}$$

Analysis of (4)–(6) shows that

$$\frac{C_{T0}J_0}{4C_{P0}} \tag{24}$$

corresponds to the maximum of propeller efficiency within the model investigated, and corresponding value of J for this case is equal to $J_0/2$. From this one can make a conclusion that for minimum energy consumption during the lift the propeller must work at the regime of maximum efficiency.

It should be noted that the experimental data (see Figures 2–8) show that maximal propeller effitiency corresponds to $J = 0.63 \div 0.65J_0$. This is due to the fact that C_P begins to decrease at $J > 0.3J_0$. On the other hand, the rate of decrease is slow enough so the difference between our model and experimental data is not high, and one can use this model at least for the prelimenary analysys.

Another thing that must be discussed is that the minimal energy consumption at hovering fase is proportional to the so called figure of merit FOM, and [3]

$$FOM \sim \frac{C_{P0}}{(C_T)^{3/2}}. \tag{25}$$

For the fixed diameter the maximal efficiency increases with the pitch increase while FOM decreases with pitch increase. Figure 1 shows this dependence for the propellers APC SP 11×3, 11×4, 11×5 [1] (see also Figures 6–8 from Appendix A).

It is interesting to compare the climb velocity with the mean velocity of the air V_A moving through the propeller area during the hovering. From (10), (11), (14), (18)

$$V = \frac{1}{b}\sqrt{\frac{T}{2a}} = \sqrt{\frac{T}{2C_{T0}\rho D^2}}J_0.$$

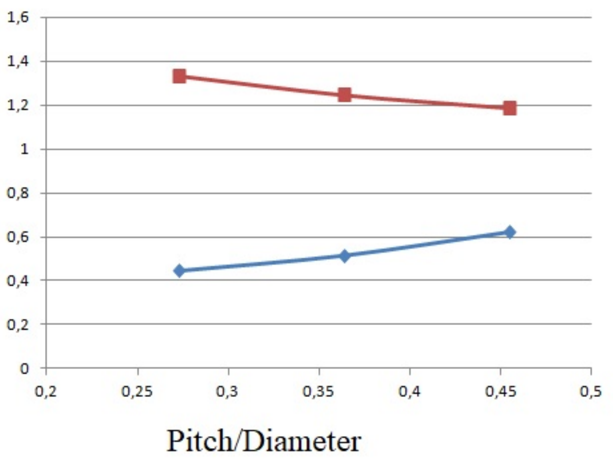


Figure 1: Efficiency (blue) and FOM (red) as functions of Pitch/Diameter.

As

$$T = \rho V_A^2 S,$$

$$S = \pi D^2/4,$$

so

$$V = V_A \sqrt{\frac{\pi}{8C_{T0}}} J_0 \simeq 0.63 \cdot V_A \frac{J_0}{\sqrt{C_{T0}}}.$$

For the propellers from Appendix A

$$V \simeq V_A.$$

This gives a simple method of optimal vertical velocity determination: measuring the air velocity past the propeller during the hovering gives the required result.

4 More precise propeller model

In Chapter 3 it was noted that the propeller model with (5) gives not very precise results. So, the idea of this chapter is to propose and test more complicated but more precise model of C_P. From Figures 2–8 in Appendix A one can see that the value of C_P decreases with the increase of J and is equal to zero at some value of J_1. On the other hand, near the zero values of J the magnitude of C_P is practically constant. From this, the idea is to approximate the dependence $C_P(J)$ by the parabola. The preliminary investigations show that the most useful form of this dependence looks like

$$C_P = C_{P0}\left(1 - \left(\frac{J}{J_0}\right)^2 (1-\delta)\right), \qquad (26)$$

where

$$\delta = 1 - \left(\frac{J_0}{J_1}\right)^2.$$

For the analysis of previous chapter $\delta = 1$.

Using the formula (26) in finding the minimum of (15) one can obtain the following:

$$n_m = \sqrt{\frac{T(\sqrt{\delta}+1)}{a\sqrt{\delta}}},$$

$$P_m = 2k\left(\frac{T}{a}\right)^{3/2}\sqrt{1+\frac{1}{\delta}},$$

$$t_m = bh\sqrt{\frac{a}{T}(\delta+\sqrt{\delta})},$$

$$E_m = 2\frac{Th}{\eta}\frac{C_{P0}}{J_0 C_{T0}}(1+\sqrt{\delta}),$$

$$\eta_{total} = \eta\frac{J_0 C_{T0}}{2C_{P0}(1+\sqrt{\delta})}.$$

The value of advanced ratio for maximal efficiency is

$$J_m = \frac{J_0}{1+\sqrt{\delta}}$$

For $\delta = 1$ these formulas become as in Chapter 3.

The value of

$$\frac{J_0 C_{T0}}{2C_{P0}(1+\sqrt{\delta})}$$

corresponds to the maximum of propeller efficiency in the model investigated. Substituting values of δ from data of Appendix A gives the corresponding value of J as $J = (0.63 \div 0.65)J_0$. These values of J and corresponding values of maximal propeller efficiencies coincide with those from graphs.

As an example let's consider the APC 11X4 propeller. According to data from [1], the coefficients for this propeller for $RPM = 6000$ are: $C_{T0} = 0.95$, $C_{P0} = 0.34$, $J_0 = 0.57$, $J_1 = 0.68$. For this data $\delta = 0.297$, $\delta^{0.5} = 0.545$, $J_m = 0.368$ $(0.64J_0)$, maximal propeller efficiency is 0.515. Data from [1] give maximal efficiency of 0.517 at $J_m = 0.369$. The dependencies of C_T, C_P and efficiency are given by red in Figure 6 with the experimental data. One can see the good coincidence between the analytical formulas and experimental data. The accuracy and deviations of analytical formulas results are mainly due to the accuracy and deviations of experimental data that are used for the determination of the coefficients.

It is also useful to analyze the case when J_1 tends to J_0. For this case δ tends to zero, and, from this, t_m tends to zero while P_m tends to infinity but their product Em tends to its minimal value (as a function of δ), rotational frequency n_m and climb velocity tend to infinity while the total efficiency tends to its maximal value with respect to δ, J_m tends to J_0. The condition $J_1 = J_2$ physically means that there is no drag on the blades of propeller at zero lift. In other words, this condition corresponds to the "best", "ideal" propeller, and the best lift scenario for this propeller is the instantaneous lift with infinite power with infinite vertical velocity. Of cause, the drag of copter's construction is not taken into account here.

Now compare the formulas with those from Chapter 3. The optimal frequency of rotation is higher than in Chapter 3 while power and time of lift are lower. More important is that the efficiency is higher than in Chapter 3. This means that the formula (23) underestimate the real value of propeller efficiency. On the other hand, for the real values of δ that are only slightly less than unity the difference in formulas is rather small.

5 MAXIMAL CLIMB ALTITUDE

For the simplification of further investigation let's rewrite the formula for the minimal energy (21) as

$$E_m = \frac{mgh}{\eta_{total}}. \tag{27}$$

The mass of accumulator m_{ac} is practically proportional to its maximal energy stored E, and proportionality coefficient is α, so

$$m_{ac} = \alpha E.$$

Assume that we can change the mass of accumulator to maximize the maximal climb altitude of the copter. If the mass of copter without accumulator mass required for climb is m_0 (m_0 can include the mass of other accumulators, required for other parts of the mission), then

$$m = m_0 + E_m. \tag{28}$$

From (27) and (28)

$$E_m = \frac{m_0 gh}{\eta_{total}} \left(1 - \frac{\alpha gh}{\eta_{total}}\right)^{-1}.$$

The maximal climb altitude corresponds to the condition of

$$h = \frac{\eta_{total}}{\alpha g}.$$

For typical LiPol accumulator the stored energy is about 200 Watt·hour/kg. For the total efficiency $\eta_{total} = 0.5$ it gives the maximal altitude of 36 km. Up to now there is no aircraft flying at such altitude. Of cause, this altitude corresponds to very heavy copter and can't be realized and we haven't take into account the aerodynamical drag of construction, the air density and temperature change with altitude, the change of construction mass with the change of total mass and many other peculiarities (that can be analyzed in the future work), but this result shows that now copters with the moderate amount of accumulators onboard can reach rather high altitudes.

6 CONCLUSION

1. On the basis of experimental data two models of propeller for the analytical investigation of minimum energy lift mode for the copters and VTOL airplanes are proposed and investigated.

2. The task of minimum energy consumption lift mode for copters and VTOL airplanes is solved. Optimal propeller parameters and regimes are obtained. It is shown that for the energy consumption minimization the propeller must work at the regime of maximal efficiency and the form of propeller must be optimized for the maximization of efficiency.

3. It is emphasized that for the minimal energy consumption during the hovering phase the propeller must work and its forms must be optimized for the maximization of Figure Of Merit value and increasing the propeller efficiency decreases its FOM.

4. The maximal available climb altitude is evaluated. It is shown that it is higher than the typical altitudes of aircraft cruise flight, so practically all the required altitudes can be reached by copter.

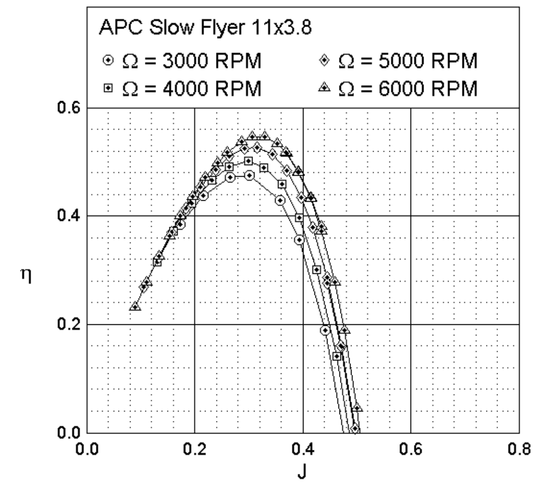
REFERENCES

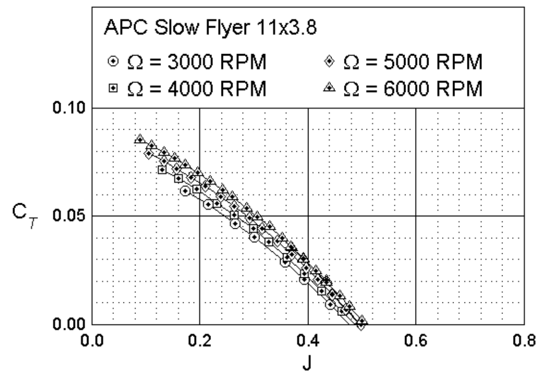
[1] https://m-selig.ae.illinois.edu/props/propDB.html

[2] *Serokhvostov S.V.* and *Churkina T.* One useful propeller mathematical model for MAV. Proc. of International Micro Air Vehicle conference and competitions 2011 (IMAV 2011), 't Harde, The Netherlands, September 12-15, 2011. Delft University of Technology and Thales, 2011

[3] *Robert W. Deters* and *Michael S. Selig* Static Testing of Micro Propellers. Materials of 26th AIAA Applied Aerodynamics Conference 18–21 August 2008, Honolulu, Hawaii. AIAA 2008–6246.

APPENDIX A: PROPELLERS DATA





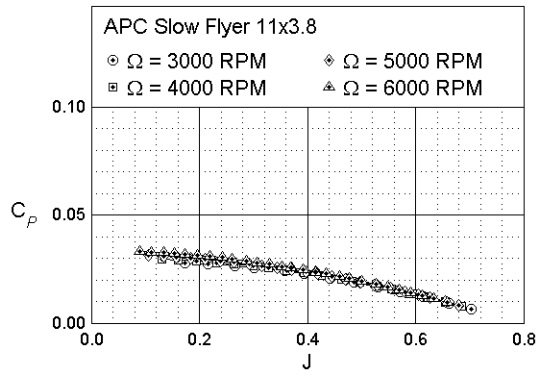
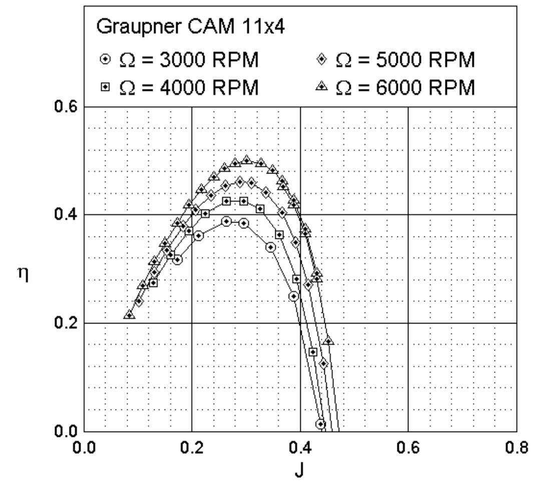
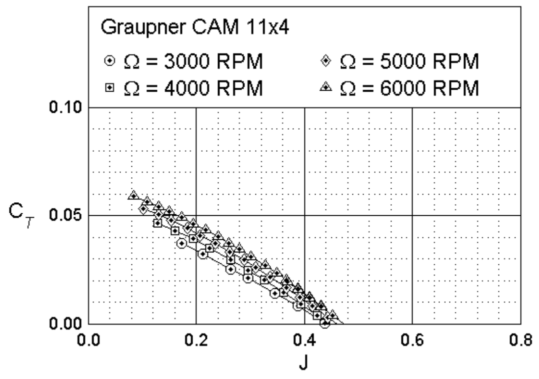


Figure 2: APC Slow Flyer 11x3.8 propeller characteristics. [1]





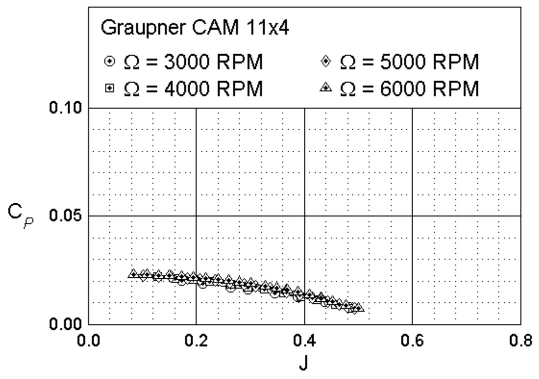


Figure 3: Graupner CAM 11x4 propeller characteristics. [1]

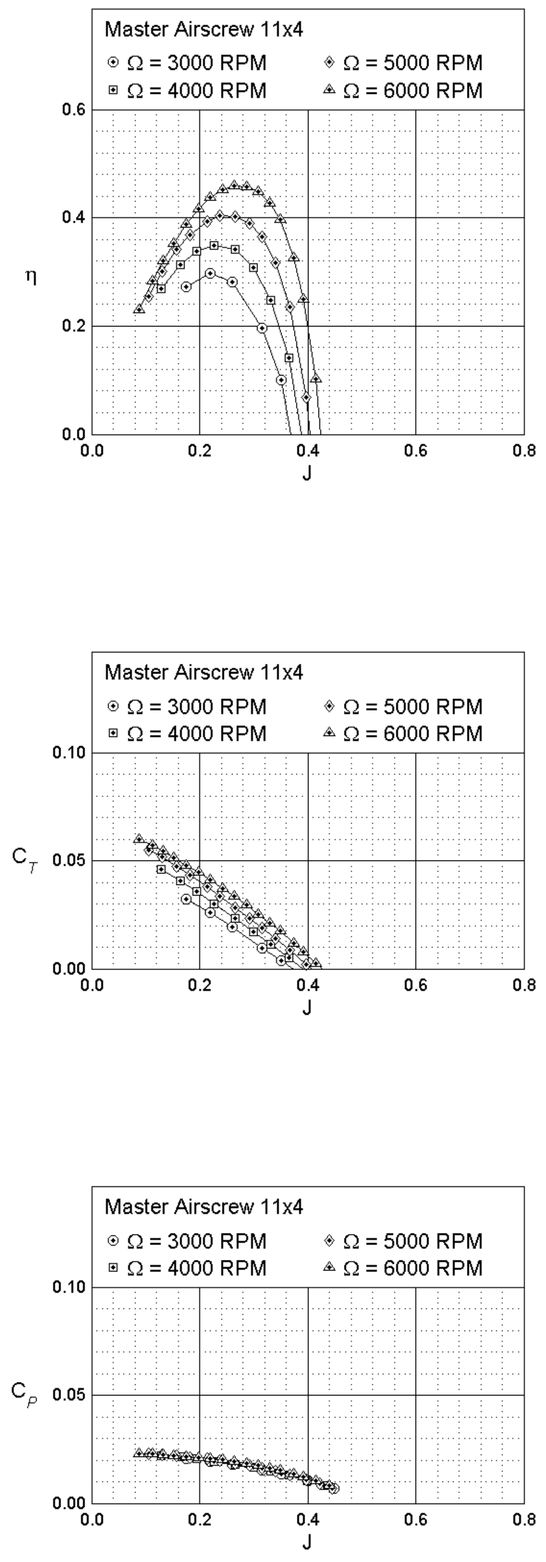


Figure 4: Master Airscrew 11x4 propeller characteristics. [1]

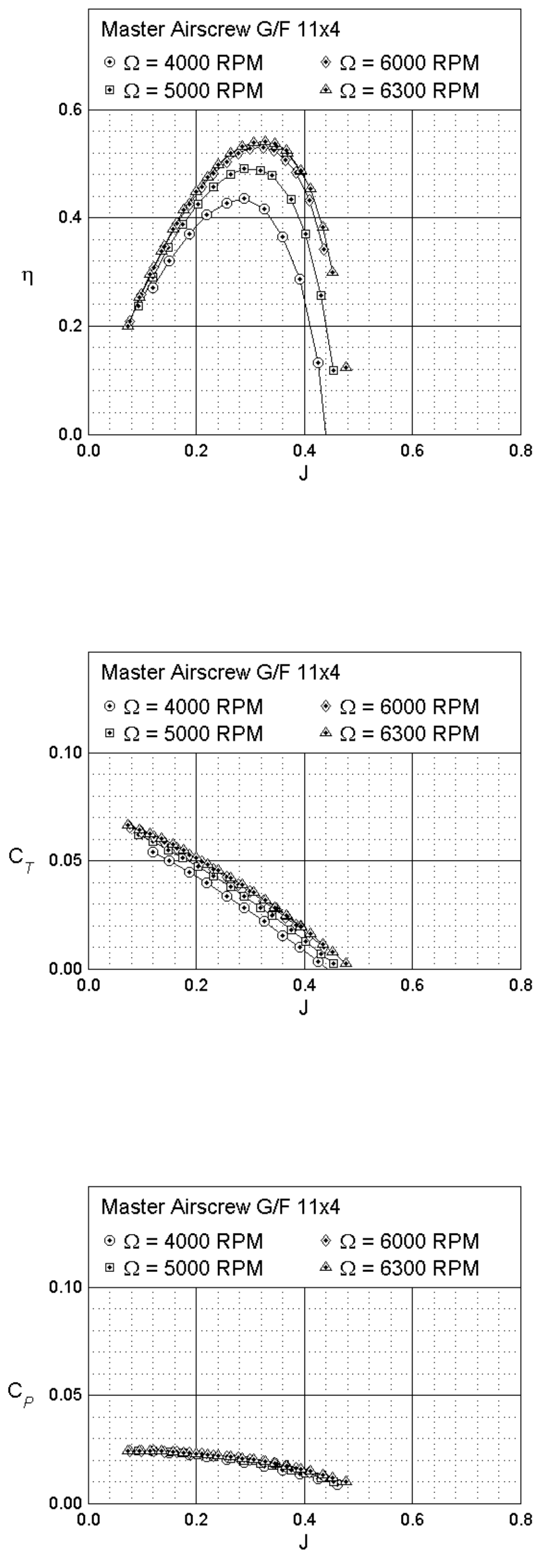


Figure 5: Master Airscrew G/F 11x4 propeller characteristics. [1]

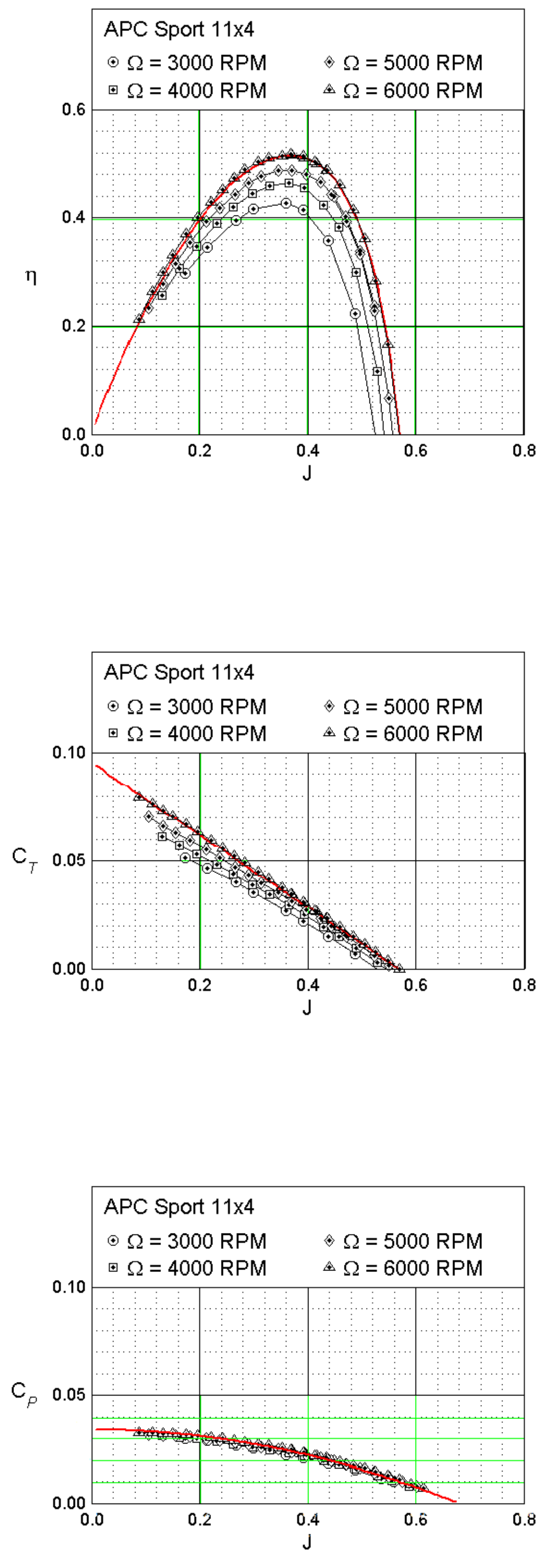


Figure 6: APC Sport 11x4 propeller characteristics [1]. Red lines are analytical formulas

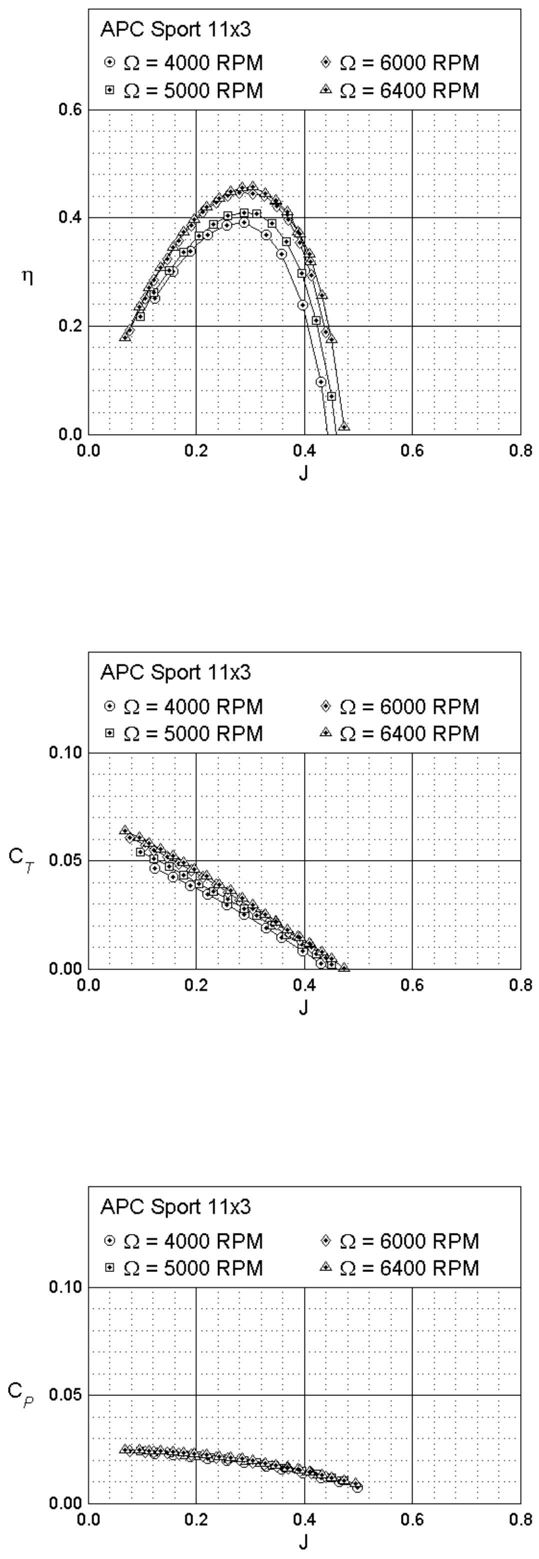


Figure 7: APC Sport 11x3 propeller characteristics. [1]

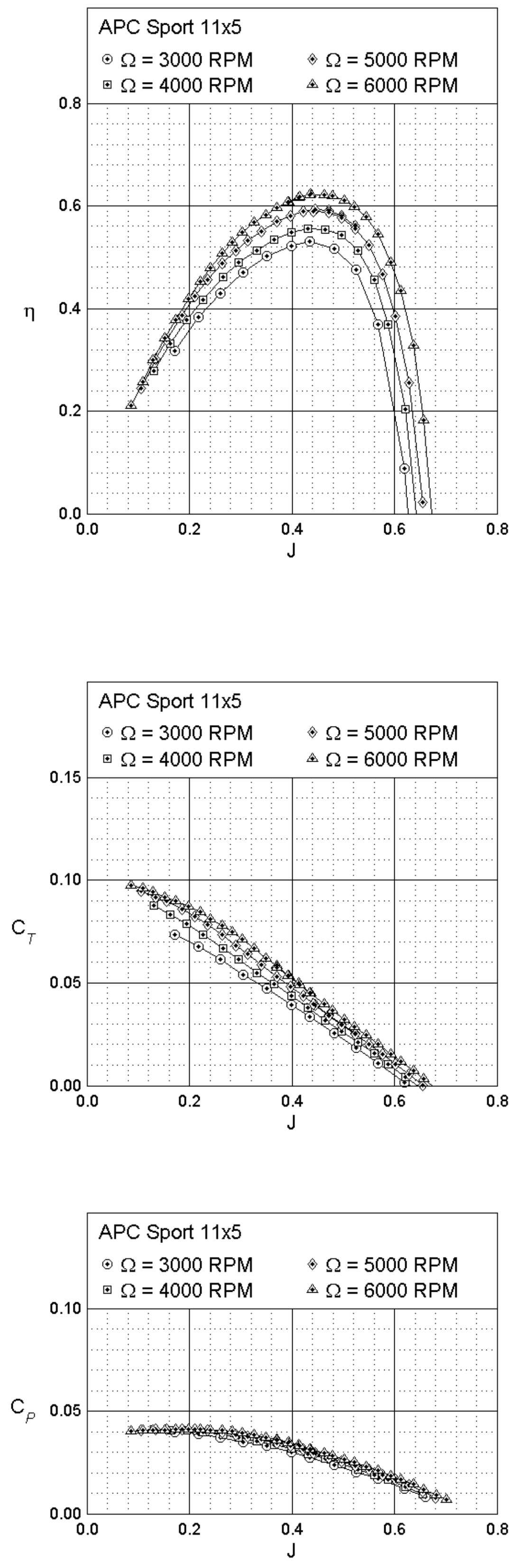


Figure 8: APC Sport 11x5 propeller characteristics. [1]

3D Reconstruction based on NIR single-pixel for drone navigation under rainy condition

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

ABSTRACT

In recent years, 3D reconstruction has become a challenging task for navigation systems. Therefore, different technologies such as RGB cameras, LIDARs, or RADAR can capture information from the environment and perform 3D reconstruction. In an environment of high dispersion and low illumination, it is necessary to have a robust solution that operates in the infrared spectrum. One solution is to integrate Single-Pixel Near Infra-Red Imaging (SPI-NIR) technology, which allows image reconstruction using few samples and operates in high dispersion conditions. In this work, an evaluation of the performance of an SPI-NIR vision system with active illumination for 3D image reconstruction in rainy environments is performed. The reconstruction of 3D objects is performed from the reconstruction of a low-resolution SPI-NIR 2D image, using a robust unified reflectance model that combines the Lambertian, Oren-Nayar, and Blinn-Phong models to improve the 3D image of objects with surface roughness or with low reflectance. For 3D reconstruction, single-view Shape-From-Shading (SFS) based on fast Eikonal solvers was used. This makes it possible to improve the shape of the 3D object, reduce computation time for future applications and generate real-time 3D images in harsh environments.

1 INTRODUCTION

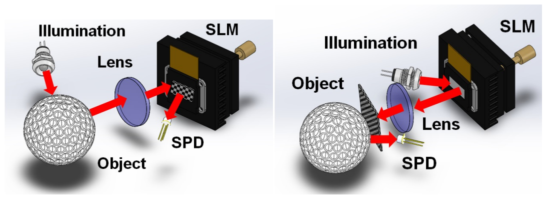
3D reconstruction is itself a challenging task. There are different type of techniques for the reconstruction as mechanical based on ultrasound [1], and the radiometric classified active through the use of Lidar [2], Time of Flight [2], interferometry [3], structured light projection [4], and passives subdivided into the single-view approaches [5], such as Shape-from-Focus(SFF), shape-from-shading(SFS) and shape-from-texture(SFT), and multi-view approaches [5], such as structure from motion and stereo-vision. The goal of a 3D surface reconstruction is to obtain depth information of an object or its surface topology. In conditions of scattering, this can be a complex task with high computational load. In the case of using RGB sensors, there are some limitations due to susceptibility to weather conditions, including rain, snow, and fog or low-visibility scenario [6]. A solution uses InGaAs sensors that operate in the near-infrared spectrum (NIR) at the wavelength 1550nm, through a system with active illumination to illuminate the surface object and make the 3D reconstruction. The NIR has a better performance in outdoor sceneries with high scattering over Long-wave infrared (LWIR) and the visible (VIS), and it can be adapted by active illumination while LWIR is not possible. However, due to the increasing droplet diameters of water particles, there is an increase in the extinction coefficient β [7] factor related to the loss of path light received for the photodetectors. This effect causes a decrease in the level contrast in the scene that can be reconstructed. In LWIR under the rainy condition, the value β is higher than the NIR spectrum. Therefore, the distance of detection is low, and there is more loss, with that the performance image reconstructed is low quite[8].

This work proposes a vision system 3D image single view based on single-pixel imaging technology that allows the 2D reconstruction using a few samples and the SFS method.The SFS 3D reconstruction method uses a single 2D image for surface reconstruction, from a specific perspective, using the changes of illumination shading to infer the 3D shape of the object,different from SFF and SFT [5] that need to have an image stack the estimate the depth. Currently, there are no reported works about the effects to apply SPI-NIR 3D reconstruction in scattering rain, for which, in this work we focus on single-view method evaluation applying shape-from-shading technique in combination with fast Eikonal solvers method [9]. Starting from a 2D single-pixel NIR low-resolution image, we performed the 3D reconstruction in a controlled scenario that simulates the rainy condition for future object detection applications on in-flight and navigation for unmanned vehicles (UMV).

2 SINGLE-PIXEL OBJECT RECONSTRUCTION

The generation of single-pixel imaging is based on the principle of spatial information modulation from the projection of a sequence of structured patterns of light through light modulation devices such as SLM, DMD, or others (see Fig 1). The object is imaged through a lens system, the intensity of the light reflected and transmitted is collected by a photodiode. The relationship between the structured and reflected light signal measurement can be depicted (1)[4].

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



(a) (b)

Figure 1: The two different approaches applied to SPI: a) Front modulation: the object illuminated by a light source and the light reflected by him gets directed through a lens onto an SLM, the light reflected is detected by a photo-detector (or single-pixel detector, SPD). b) Back modulation: the SLM device projects a sequence of patterns and the reflected light is capture by the SPD[4].

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

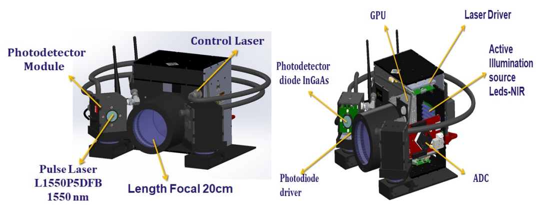
Where (x, y) is the spatial coordinate, O denotes object reflectivity, Φ_i is i_{th} structured pattern, S_i is the i_{th} single-pixel measurement corresponding to Φ_i, α is a constant factor depending on the opto-electronic response of the used single-pixel detector, and denotes a point-wise multiplication. The size of both the object and the pattern is M × N pixels, and through the knowledge of the structured patterns and reflected light signal measurement is possible to apply an algorithm to recover the object image. The reconstructed image I is proportional to object reflectivity O. The reconstructed object image can be depicted (2)[4].

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

Where the reconstructed image is expressed as the inner product of the measurements and the structured pattern, the spatial light modulation in single-pixel imaging can be implemented in two different schemes [4], a structured illumination scheme termed front modulation (Fig.1a) and a structured detection scheme termed back modulation (Fig.1b).

2.1 SPI-NIR Vision system test architecture

In this work, We propose a vision system SPI-NIR of the type back modulation (see figure 1b) with an active illumination through a NIR LEDs 8x4 matrix, in the range of the 1550 nm wavelength.The back modulation configuration offers the advantage that light is captured directly by the SPD, and it doesn't need to adapt a light divider and lens internally to concentrate the light over the SPD as front modulation(see figure 1a) with reducing the weight and the complexity of the system.The active illumination offers the capacity to project patterns in low-vision conditions (scenarios with dust, fog, rain, or smoke), increasing the capacity of outdoor operation. Another advantage is the fact that the atmosphere being able to absorb the wavelength between 1500-1600 nm [10], for which our vision system SPI is less sensitive to background noise [11], increasing the range of the detection in outdoor conditions, which gives an operating advantage for future applications as a vision sensor for applications of navigation in unmanned Aerial Vehicle (UAV) (see Fig.3). The SPI-NIR architecture proposed in this work is divided into two stages: the first will control the elements used to generate images through the single-pixel principle: a photo-detector (diode FGA015 @ 1550nm), source light, and ADC (see Fig.2a), and a second stage is the responsible of processing the signal captured by the photo-diode module through the use of an ADC, which is controlled by the GPU unit (see Fig.2b). The GPU unit (Jetson Nano) is also responsible for generating the Hadamard patterns and processing the converted data by the ADC, used by the Batch-OMP[12] algorithm running in the GPU unit to generate the 2D/3D image.To design the vision system, we define some parameters performance for integration as sensor vision for UAVs. In image capture under rainy conditions, we define exposure time as taking one SPI image in 1/50 s, for an aperture to f/2.38 with an exposure value EV=-4 and an ISO of 5000 sensitivity[13], with we can take the image outdoor with low-visibility. Others effect consider in our design is the motion blur <20% , for which we define flight speed maximum between 7.5 m/s to 30m/s for different flight height (see Table.1).


Photodetector Module
Control Laser
Pulse Laser L1550P5DFB 1550 nm
Length Focal 20cm
GPU
Laser Driver
Photodetector diode InGaAs
Active Illumination source Leds-NIR
Photodiode driver
ADC

(a) (b)

Figure 2: The proposed vision system dimension is 11x12x13 cm, focal length 20 cm, weight 1.3kg, power consumption 25W, a) first stage module photodiode, active illumination source, photodetector diode InGaAs FGA015, b) second stage GPU unit and ADC[14].

Table 1: Effects of fight height on ground sampling distance (GSD) and fight speed for a motion blur 20% and a shutter speed of 1/50s.

Flight height(m)	10	20	30	40
Flight speed(m/s)	7.5	15	22.5	30
GSD(mm)	0.75	1.5	2.25	3

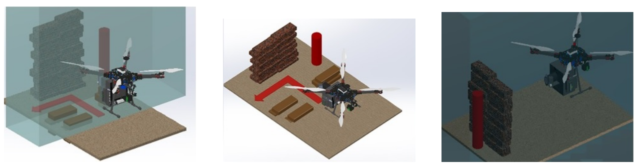


Figure 3: Vision sensor SPI-NIR for applications in UMV, a) autonomous Navigation in adverse environments, b)obstacle detection, c)3D reconstruction of scenes with low-illumination [14].

3 Scattering effect

When light interacts with water particles in the air, various physical phenomena occur as reflection, refraction, absorption, and scattering. This interaction produces dissipation of the photon's energy, the phenomenon is called scattering. The scattering is related to the size particles $sp = 2\pi r/\lambda$ of the particle [15], r being the radius. According to the Rayleigh scattering model, the light scattering strongly dependent on the wavelength and to be inversely proportional to the fourth power of the wavelength [6]. However, the particles in the atmosphere exhibit some different sizes. A more realistic relationship between extinction efficiency and size particle size approximates the probability of scattering to occur when the wavelength of the incident radiation is approximately equal to the particle's radius. Thereby, small particles with a radius less than 1mm scatter mainly in the visible portion of the spectrum, and particles of more significant size scatter stronger in the spectrum IR [6].

3.1 Modeling rain environment

We proposed a rain model adapted to single-pixel to determine the attenuation factor of the source light $\alpha\left(\lambda, D\right)$ for different drop sizes and determinate the maximum distance for possible measurements (seen figure 7 Appendix A). The proposed model consisted of three parts, first we define the weather parameters in rain environments as the reflectivity (Reflectivity factor) $Z = \sum N_i D^6$ [16], Z is related with level concentration rain N_o[17], and the differential reflectivity Z_{dr} depend on the rain rate defined as $dBZ_{dr} = 10log(Z_h/Z_v)$ [18], where Z_h horizontal polarized reflectivity [18] and Z_v vertical polarized reflectivity [18]. These parameters depend on drop diameter D, which is given by the Marshall Palmer distribution[19]. In the second part, we considered the Mie scattering effect for which we calculate the scattering efficiency Q_{sca}[20], absorption efficiency Q_{abs} [20], and extinction efficiency Q_{ext} [20] for a particle with diameter D and refractive index m, Q_{abs}[20] is related to an absorption coefficient $\mu_a\left(\lambda, D\right)$[20] and ,the third part of the model we considered the rain speed effect [21]. In the rainy condition, the effective measurement range is reduced due to the number of photons $E(N)$[14] impinging the photo-detector photo-active is less, varying the range of measurement from 18 cm to 10 cm for materials with a reflection index of 0.2 and the 28.5 cm to 12.6 cm and, for materials with a reflection index of 0.8 for droplet size $0.5mm$ (seen table 2). This variation in the measurement range caused by scattering should be considered for the 3D reconstruction test.

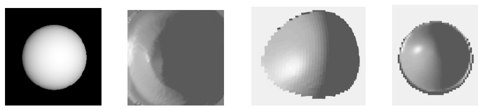


Figure 4: Generation 3D image through of rectance model, a) 2D image, b)3D image lamberting model, c)3D image Oren-Nayar model,d) 3D image Unified model.

Table 2: Single-pixel maximum measured distance.

Reflection index	0.2	0.5	0.8
Dry maximum distance (cm)	18.4	22.4	24.4
Rain dropping size @3 mm (cm)	16	18	19.6
Rain dropping size @2 mm (cm)	15.6	17.2	18.8
Rain dropping size @1 mm (cm)	14	16	17.2
Rain dropping size @0.5 mm (cm)	10	11.2	12.6

4 Reflectance model

In previous works have been reported different reflectance models, for reconstruction of the 3D image using the method of SFS, as Lambertian [22], Oren–Nayar [23] and Blinn–Phong [24]. Each model has an own feature to be applied for different types of surfaces from smooth to rough or diffuse, for which in this work, we unified a reflectance model through the linear combination of the models previously mentioned.

4.1 Lamberting reflectance model

A Lambertian surface has the property of invariant luminance according to the viewing angle (see Fig.4b). Lambert's law determines how much of the incident light is reflected, which is constant in any direction, which means that the reflected intensity is not dependent on the viewing angle but the light source's orientation relative to the surface is. Therefore, the Lambertian surface is modeled as the light source intensity I_o product, the surface albedo ρ, and the cosine of the angle θ, between source directions S and surface normal N (3)[22]. The Lambertian model is very straightforward and computationally efficient, widely applied in the reconstruction of 3D images.

$$I = \frac{I_o}{\pi}\rho cos\left(\theta\right) \tag{3}$$

4.2 Oren–Nayar and Blinn–Phong Reflectance Model

Due to the Lambert model is based on smooth surface reflectance, in diffuse surface conditions, the Oren-Nayar model improves 3D reconstruction, considered the roughness of the surface as the standard deviation of the Normal Gaussian distribution (see Fig.4c)(4) [23]

$$L_r(\theta_i, \phi_i, \theta_r, \phi_r) = I_o \frac{\rho_d}{\pi} cos\phi_i (A + B \\ max[0, cos(\theta_r - \theta_i)] \, sin\alpha tan\beta \quad (4)$$

where $A = 1 - 0.5\sigma^2/(\sigma^2 + 0.33)$, $B = 0.45\sigma^2/(\sigma^2 + 0.09)$, $\alpha = max[\theta_i, \theta_r]$, $\beta = min[\theta_i, \theta_r]$

An improvement of the Oren-Nayar model is the Blinn-Phong model which includes an ambient term and it is based on that shiny surfaces have small intense specular highlights, while dull surfaces have significant highlights that fall off more gradually (5) [24].

$$L_r = \omega_d I_o \frac{\rho_d}{\pi} cos\theta_i + \omega_s I_o \frac{\rho_s}{\pi} \left(\frac{R}{\|R\|} \cdot \frac{V}{\|V\|} \right)^n \quad (5)$$

4.3 Unified Reflectance Model

Due to the Lambertian model is inaccurate in rough diffuse surfaces, we proposed to combine through a linear combination of Oren–Nayar model and the specular part of Blinn–Phong model (see Fig.4d)(6)[9].

$$L_r = \omega_d I_o \frac{\rho_d}{\pi} cos\phi_i (A + B \\ max[0, cos(\theta_r - \theta_i)] \, sin\alpha tan\beta + \omega_s I_o \frac{\rho_s}{\pi} (n \cdot h)^n \quad (6)$$

5 3D Image Reconstruction

The success of 3D image reconstruction based on the method of SFS depends greatly on the physical brightness surface and reflectance properties as well as the illumination conditions, and other extrinsic factors as camera properties (see Appendix B) and the reference coordinate system of the viewer/camera-centered. This method seeks to solve the image irradiance equation $E_r = R(n(x))$[25] where the normalized brightness $I(x)$ of gray-value image is related with the reflectance map $R(n(x))$. For 3D image reconstruction, we create a photo-metric image from a 2D low-resolution reconstructed image using a single-pixel with active illumination based on a Leds NIR array of 8x4 in the wavelength of 1550 nm in a rainy scenario (seen figure 5). The fact that our visual system has illumination active allows keeping a level of brightness not depend on the condition of the background light and rainy condition or scattering medium in the near-infrared present less attenuation than vision systems that work in the visible spectrum [6]. Using the 2D SPI-NIR image, the irradiance level is estimated E_r. Calculating the parameters albedoρ_d, θ_r and θ_i must be considering that a single light source of one direction L(sequence projection light using Hadamard pattern) with the image plane that coincides with the optical z-axis of the camera (detector single pixel), was used. Due to the relationship between the surface reflected radiance model L_r and the irradiance image E_r (2D image SPI-NIR), we can define $E_r = \eta L_r$[25] with that we can define irradiance image with information of depth (7).

$$I(x, y) = cos\phi_i (A + B \\ max[0, cos(\theta_r - \theta_i)] \, sin\alpha tan\beta + (n \cdot h)^n \quad (7)$$

For solving (7), we can consider that the direction of the light source is the same as the camera, with that $\theta_i = \theta_r, \phi_i = \phi_r, \alpha = \theta_i = \beta$ and, $n \cdot h = cos(\theta_i)$, we can express irradiance image (9) [26] (see appendix B).

$$I(x, y) \sqrt{1 + \|\triangledown z(x, y)\|^2} + \tilde{\omega} \cdot \triangledown z(x, y) \\ - \omega_s = 0, in\Omega \quad (8)$$

if we consider that source has one direction $\omega = (0, 0, 1)$ and (8) , it can be define as Eikonal equation [26](9).

$$|\triangledown z(x, y)| = f(x), wheref(x) = \sqrt{\frac{1}{I(x, y)^2} - 1} \quad (9)$$

6 Experiment and Results

To evaluate the 3D reconstruction vision system's capabilities with active illumination through the method of SFS with Eikonal solver (9) in rain condition, we develop a testing bench that has a controlled system of illumination to simulate background outdoor light and a system that can simulate the conditions of rain with a size drop of 0.5 mm,1 mm, and 2 mm (see Fig. 5.). For the 3D image reconstruction initially, we take a 2D image single-pixel NIR of low-resolution, which is obtained from 80% of the Hadamard patterns projected, and it is reconstructed through the technique of CS using the Batch algorithm in combination with the method of FSRCNN with upscaling factors of 4 [27] (see Fig.6 a,d). Due to the scattering effect caused by rain, a pre-process of the 2D SPI-NIR image was needed through Gaussian filter and morphologic methods before making the 3D image reconstruction using SFS. In this work, we focus on reconstructing objects with few details on their shape with rough surfaces in rainy conditions. For the first test, we reconstructed a homogeneous surface, for which we chose a bright spherical (a shape sphere object can be an ideal approximate of the Lambert surface and can be considered as a calibration object in the 3D reconstruction[25]) object of 50mm of diameter at a distance of between 10 to 17 cm from the focal lens (see Fig.6a) in which we applied scanning of the type basic and spiral. In the second test we reconstructed a rectangular area formed by a cube, its area is 40x40x40mm at a distance between 10 to 17 cm from the focal lens (see Fig.6d), and we applied a

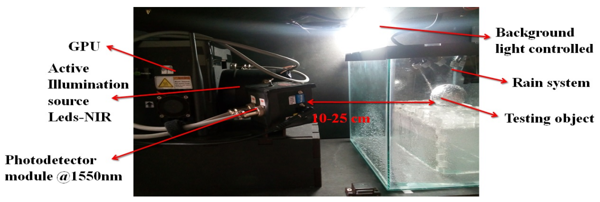

GPU
Active Illumination source Leds-NIR
Photodetector module @1550nm
10-25 cm
Background light controlled
Rain system
Testing object

Figure 5: Experimental setup to vision system 3D SPI-NIR, the testing bench has a rain (size drop rain 0.5,1,2 mm) and background light-controlled system. Range of measuring between 10-25 cm (seen table I), the testing object must be positioned into the glass box to evaluate the performance of the 3D image reconstruction in rain conditions.

scanning basic and ZigZag. for the test, we used different scanning methods to determine which method can adapt better to rainy conditions depended on the type of surface that want to reconstruction.

6.1 *Discussion: Testing 3D SPI-NIR reconstruction in rainy condition and without rain*

In the tests carried out to 3D image reconstruction in rainy conditions, we define the measured distance between 10 to 17 cm, from the analysis of the modeling the SPI-NIR in condition rain (see table 2), the distance that is in the range of measure of an object with a high reflection index. In rainy conditions, for critical droplet sizes $< 1mm$ where the scattering effects are more, we can see a high level degradation of the quality of both objects' reconstructed 2D/3D image, the spherical object representing a more significant degradation. For the object, the square can see a degradation in the surface reconstructed due to attenuation and loss of information (see Fig.6f). If we applied a scanning method of the type Zig-Zag, it can improve the quality of 2D/3D image with a better level, $SSIM > 0.7$, compared to the basic scanning method. In the case of the reconstruction of the object with sphere shape in the test, we can see that the scanning method spiral have a better level SSIM in the rainy condition, and the 3D image has a high-level degradation for droplet sizes $0.5mm$ with a more significant loss of information (see Fig.6b,c) in comparison with a square object.About the processing time for reconstruction of sphere objects, the time is between 28 to 32 ms with the Hilbert scan as the lower, and Spiral as the faster, for case square object the time is between 27 to 30 ms, with the Basic scan being lower, and Zig-Zag is faster.

7 Conclusion

This paper presented a vision system SPI-NIR with active illumination of low resolution to 3D image reconstruction in rainy conditions for future application on in-vehicle UAVs. The 3D image reconstruction is based on the single-view methods shape-from-shading(SFS) with an Eikonal solver. In this work, first, we defined the measurement ranges by modeling the number of photons detected in the rainy condition (seen Appendix A Fig.7), and we can determine the effective measurement range between 10-28 cm (see table 2). Due to the scattering effect caused by the rain, the level of light reflected off the object will be attenuate, for which we proposed using a unified reflection model (6) to estimate 2D SPI-NIR image irradiance used to make the reconstruction 3D. In the test, we can see as the droplet size affect the quality of the 2D/3D image reconstructed. In the condition of fine drizzle with size droplet $< 1mm$, the scattering effect is more significant, so the 3D reconstruction has limitations. We implement different scanning methods for this as a solution depending on the type of element to detect. For example, in an urban ambient with many buildings, the 3D reconstruction is focused on detecting walls that can be approximate the reconstruction of an object with a shaped square. So the scanning ZigZag is a method more the most appropriate to make a 3D reconstruction. In contrast, in a scenario with more vegetation or mountains, we can approximate some scenario objects as shape spherical or curves, being a scanning Spiral method the most appropriate. This capacity to improve the quality of the 3D image increases the vision system's capabilities to applications to vehicles UMV in scenarios with rain.

References

[1] Qinghua Huang and Zhaozheng Zeng. A review on real-time 3d ultrasound imaging technology. *BioMed Research International*, 2017:6027029, Mar 2017.

[2] Maged Aboali, Nurulfajar Abd Manap, Abd Darsono, and Zulkalnain Yusof. Review on three dimensional (3-d) acquisition and range imaging techniques. *International Journal of Applied Engineering Research*, 12:2409–2421, 06 2017.

[3] Fabian Wagner, Florian Schiffers, Florian Willomitzer, Oliver Cossairt, and Andreas Velten. Intensity interferometry-based 3d imaging. *Opt. Express*, 29(4):4733–4745, Feb 2021.

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

[5] Richard Hartley and Andrew Zisserman. *Camera Geometry and Single View Geometry*, page 151–152. Cambridge University Press, 2 edition, 2004.

[6] M. I. Mishchenko, L. D. Travis, and A. A. Lacis. *Multiple Scattering of Light by Particles: Radiative Transfer and Coherent Backscattering*. Cambridge University Press, Cambridge, 2006.

[7] J. Wojtanowski, M. Zygmunt, M. Kaszczuk, Z. Mierczyk, and M. Muzal. Comparison of 905 nm and 1550 nm semiconductor laser rangefinders' performance deterioration due to adverse environmental conditions. *Opto-Electronics Review*, 22(3):183–190, Sep 2014.

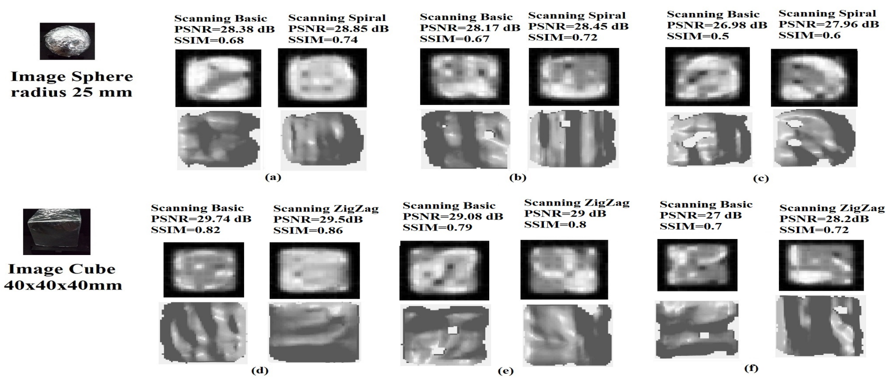


Image Sphere radius 25 mm
Scanning Basic PSNR=28.38 dB SSIM=0.68
Scanning Spiral PSNR=28.85 dB SSIM=0.74
(a)
Scanning Basic PSNR=28.17 dB SSIM=0.67
Scanning Spiral PSNR=28.45 dB SSIM=0.72
(b)
Scanning Basic PSNR=26.98 dB SSIM=0.5
Scanning Spiral PSNR=27.96 dB SSIM=0.6
(c)
Image Cube 40x40x40mm
Scanning Basic PSNR=29.74 dB SSIM=0.82
Scanning ZigZag PSNR=29.5dB SSIM=0.86
(d)
Scanning Basic PSNR=29.08 dB SSIM=0.79
Scanning ZigZag PSNR=29 dB SSIM=0.8
(e)
Scanning Basic PSNR=27 dB SSIM=0.7
Scanning ZigZag PSNR=28.2dB SSIM=0.72
(f)

Figure 6: 3D SPI-NIR Image reconstruction with active illumination in wavelength of 1550 nm of a spherical object of 50mm diameter and cube object of 40x40x40mm both at a distance between 10 to 17 cm from the focal lens with a processing time of 28 ms and 32 ms respectively, we scanning basic and spiral to evaluate performance, we calculate the parameters PSNR and SSIM, a,d) without rain at 17 cm,b,e)in rainy conditions with a size drop diameter of 2mm at 15 cm,c,f) in rainy conditions with a size drop diameter of 0.5mm at 10 cm.

[8] Dragana Perić, Branko Livada, Miroslav Perić, and Saša Vujić. Thermal imager range: Predictions, expectations, and reality. *Sensors*, 19(15), 2019.

[9] Guohui Wang, Xuan Zhang, and Jin Cheng. A unified shape-from-shading approach for 3d surface reconstruction using fast eikonal solvers. *International Journal of Optics*, 2020:6156058, May 2020.

[10] Solar radiation photosynthetically active radiation. https://www.fondriest.com/environmental-measurements/parameters/weather/photosynthetically-active-radiation/. Accessed: 2021-01-28.

[11] R. Lange, S. Böhmer, and B. Buxbaum. 11 - cmos-based optical time-of-flight 3d imaging and ranging. In Daniel Durini, editor, *High Performance Silicon Imaging (Second Edition)*, Woodhead Publishing Series in Electronic and Optical Materials, pages 319–375. Woodhead Publishing, second edition edition, 2020.

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

[13] Lukas Roth, Andreas Hund, and Helge Aasen. Phenofly planning tool: flight planning for high-resolution optical remote sensing with unmanned areal systems. *Plant Methods*, 14(1):116, Dec 2018.

[14] Carlos Alexander Osorio Quero, Daniel Durini Romero, Ruben Ramos-Garcia, Jose de Jesus Rangel-Magdaleno, and Jose Martinez-Carranza. Towards a 3d vision system based on single-pixel imaging and indirect time-of-flight for drone applications. In *2020 17th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, pages 1–6, 2020.

[15] Kuo-Nan Liou. An introduction to atmospheric radiation / kuo-nan liou., 1980.

[16] Matthew Lebsock and Tristan L'Ecuyer. The retrieval of warm rain from cloudsat. *Journal of Geophysical Research*, 116, 10 2011.

[17] Xiantong Liu, Qilin Wan, Hong Wang, Hui Xiao, Yu Zhang, Tengfei Zheng, and Lu Feng. Raindrop size distribution parameters retrieved from guangzhou s-band polarimetric radar observations. *Journal of Meteorological Research*, 32(4):571–583, Aug 2018.

[18] S-G. Park, V. N. Bringi, V. Chandrasekar, M. Maki, and K. Iwanami. Correction of radar reflectivity and differential reflectivity for rain attenuation at x band. part i: Theoretical and empirical basis. *Journal of Atmospheric and Oceanic Technology*, 22(11):1621 – 1632, 01 Nov. 2005.

[19] J. S. Marshall and W. Mc K. Palmer. The distribution of raindrops with size. *Journal of Atmospheric Sciences*, 5(4):165 – 166, 01 Aug. 1948.

[20] Jing Guo, He Zhang, and Xiang-jin Zhang. Propagating characteristics of pulsed laser in rain. *International Journal of Antennas and Propagation*, 2015:292905, Sep 2015.

[21] Ross Gunn and Gilbert D. Kinzer. The terminal velocity of fall for water droplets in stagnant air. *Journal of Atmospheric Sciences*, 6(4):243 – 248, 01 Aug. 1949.

[22] Matt Pharr, Wenzel Jakob, and Greg Humphreys. 08 - reflection models. In Matt Pharr, Wenzel Jakob, and Greg Humphreys, editors, *Physically Based Rendering (Third Edition)*, pages 507–568. Morgan Kaufmann, Boston, third edition edition, 2017.

[23] Michael Oren and Shree K. Nayar. Generalization of the lambertian model and implications for machine vision. *International Journal of Computer Vision*, 14(3):227–251, Apr 1995.

[24] Bui Tuong Phong. *Illumination for Computer Generated Pictures*, page 95–101. Association for Computing Machinery, New York, NY, USA, 1998.

[25] Fabio Camilli and Silvia Tozza. A unified approach to the well-posedness of some non-lambertian models in shape-from-shading theory, 2016.

[26] Silvia Tozza and Maurizio Falcone. Analysis and approximation of some shape-from-shading models for non-lambertian surfaces, 2016.

[27] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. *CoRR*, abs/1608.00367, 2016.

[28] Laser safety facts. https://www.lasersafetyfacts.com /laserclasses.html. Accessed: 2021-05-28.

[29] K. Garg and S.K. Nayar. Detection and removal of rain from videos. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I, 2004.

[30] Nicolas Hautière, Jean-Philippe Tarel, and Didier. Free space detection for autonomous navigation in daytime foggy weather. 05 2009.

[31] Emmanuel Prados and Olivier Faugeras. A generic and provably convergent shape-from-shading method for orthographic and pinhole cameras. *International Journal of Computer Vision*, 65(1):97–125, Nov 2005.

APPENDIX A: MODELING RAIN

A.1 Modeling SPI-NIR rain condition

The vision system proposed in this work is SPI-based [4], with illumination active for which we define as source light array LEDs of 8x4 that work in the wavelength of 1550 nm with a power level of 3.2W (IEC Eye Safety regulation IEC62471 [28]). The SPI-NIR system's critical factor is the minimum integration time T_{int} required by the photo-diode to capture the photons emitted by the array of LEDs to convert them into an electrical signal and above the background noise. This factor T_{int} determines the measuring maximum to reach outdoor by using (9)[14] that define the number of photons E(N) impinging the photo-detector photo-active in dry condition, and it depends on the spectral content $1544nm < \lambda < 1556nm$, the detector quantum efficiency $QE\left(\lambda\right)$ in this bandwidth, the length of the integration time of the detector T_{int}, and pixel's effective photo-sensitive area A_{pix}, defined as $A_{wxl}FF$, where A_{wxl} is the semiconductor window and FF the photodiode's fill-factor, $\Phi_{e\lambda}$ defined as the irradiation level of the active source, $E_{e\lambda\_sum(\lambda)}$ irradiation level of the sun illumination considered to be of 100 Klux, the $f_{\#}$ number define as $f_{\#} = f_{foc}/d_{aperture}$, f_{foc} is the length focal , and $d_{aperture}$ opening distance is the focal distance/opening distance, h is Planck's constant= $6.62607004x10^{-34}m^2kg/s$, z is the measured distance, c is the speed of light constant,τ the lens transmittance,ρ the material reflection index, and α_{FOV} the focal aperture angle [14] of the emitting LED array.

$$E(N) = \int_{\lambda_1}^{\lambda_2} \frac{\rho\tau_{lends}QE\left(\lambda\right)T_{int}A_{pix}FF\lambda}{hcf_{\#}^2} \left[E_{e\lambda\_sum(\lambda)} + \frac{\Phi_{e\lambda}}{\pi z^2 tan\left(\alpha_{FOV}\right)}\right] d\lambda \quad (10)$$

In rain condition, the irradiation level of the active source illumination is attenuate due to the scattering coefficient $\mu_s\left(\lambda, D\right)$ (11)[20] and absorption coefficient $\mu_a\left(\lambda, D\right)$(12)[20] relation of scattering Mie and weather parameters of rain environment model(seen figure 6) $\alpha\left(\lambda, D\right)$[20]. For which the distance of measure of the SPI system in scattering environment is less than to dry conditions (seen table I).To evaluate the effective measured range theoretical the (10) must consider the variables of the rain environment model proposed as $\alpha\left(\lambda, D\right)$[20], speed rain (13)[29] , the brightnesses of the dropping rain L_{r_i} and light of the background. Factors that affected the number of photons E(N) impinging the photodetector photoactive. The irradiation level of the active source (array LEDs) $\Phi_{e\ lambda}$ is attenuate a factor $\Phi_{e\lambda}e^{-\alpha(\lambda,D)z}$ and the $E_{e\lambda\_sum(\lambda)}$ of level background illumination is defined as $E_{e\lambda\_sum(\lambda)} = \left\lfloor \tau_{rain}AL_r + \tau_{rain}\left(1 - A\right)L_b + \left(T_{patterns} - \tau_{rain}L_b\right)\right\rfloor$ [30] ,where $L_{r_i} = L_{r_{i-1}}e^{-\beta z} + L_b\left(1 - e^{-\beta z}\right)$ [30] and the brightness level of the background light received by the detector is affected by the speed of the raindrops and the projection time of the projected Hadamard light patterns, as a gain factor of the received light we have the variable $A = \frac{\pi f_{foc}^2 D^2}{z}$ which corresponds to focal parameters of our system, the β variable is defined as a geometric factor of the raindrop $0 < \beta < \frac{\sqrt{D}}{50T_{patterns}}$[29].

$$\mu_s\left(\lambda, D\right) = \int \sigma_s\left(\lambda, D\right)N(D)dD \quad (11)$$

$$\mu_a\left(\lambda, D\right) = \frac{\pi D^2}{4}Q_{abs}\left(\lambda, D\right) \quad (12)$$

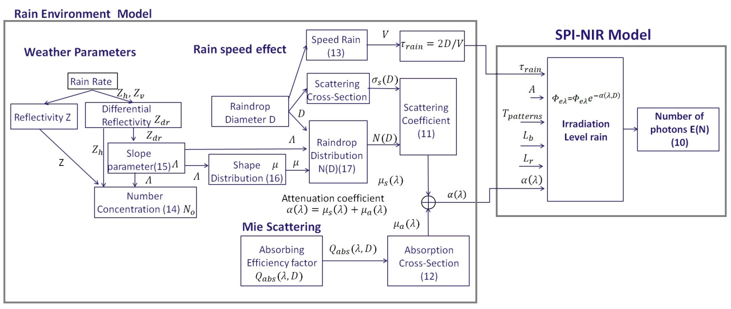

Rain Environment Model
Weather Parameters
Rain speed effect
Rain Rate
Z_h, Z_v
Reflectivity Z
Differential Reflectivity Z_{dr}
Z_{dr}
Z_h
Z
Slope parameter(15) Λ
Λ
Number Concentration (14) N_o
Raindrop Diameter D
D
Speed Rain (13)
V
$\tau_{rain} = 2D/V$
Scattering Cross-Section
$\sigma_s(D)$
Scattering Coefficient (11)
Raindrop Distribution N(D)(17)
$N(D)$
Shape Distribution (16) μ
μ
$\mu_s(\lambda)$
Attenuation coefficient $\alpha(\lambda) = \mu_s(\lambda) + \mu_a(\lambda)$
$\alpha(\lambda)$
$\mu_a(\lambda)$
Mie Scattering
Absorbing Efficiency factor $Q_{abs}(\lambda, D)$
$Q_{abs}(\lambda, D)$
Absorption Cross-Section (12)
SPI-NIR Model
τ_{rain}
A
$T_{patterns}$
L_b
L_r
$\alpha(\lambda)$
$\Phi_{e\lambda}=\Phi_{e\lambda}e^{-\alpha(\lambda,D)}$
Irradiation Level rain
Number of photons E(N) (10)

Figure 7: Overall block diagram single-pixel rain environment model proposed, we considered weather parameters, Mie Scattering,absorption efficiency Q_{abs}[20], speed dropping rain,raindrop distribution $N(D)$[20], concentration rain N_o[17], slope concentration distribution Λ[17],shape parameter μ[17], and SPI-NIR model to evaluate the range detection practical theoretical of the detector InGaAs (1550 nm).

$$v = -0.1021 + 4.932D - 0.9551D^2 + 0.07934D^3 - 0.002362D^4 \quad (13)$$

$$N_o = Z_h 10^{\left(0.00285Z\Lambda^3 - 0.0926\Lambda^2 + 1.409\Lambda - 3.764\right)} \quad (14)$$

$$\Lambda = 0.0125{Z_{dr}}^{-3} - 0.3068{Z_{dr}}^{-2} + 3.3830{Z_{dr}}^{-1} + 0.179 \quad (15)$$

$$\mu = -0.0201\Lambda^2 + 0.902\Lambda - 1.718 \quad (16)$$

$$N(D) = N_0 D^{\mu} e^{-\Lambda D} \quad (17)$$

Appendix B: Intrinsic Matrix and Eikonal Solvers

B.1 Intrinsic Matrix

The intrinsic matrix of the vision system SPI is defined using the model Pinhole with a focal length 25 cm,and axis skew, s=0 (18)[31].

$$K = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 114 & 0 & 90 \\ 0 & 114 & 100 \\ 0 & 0 & 1 \end{bmatrix} \quad (18)$$

B.2 Unified Eikonal-Type PDE Equation

For method, SFS we define a 3D surface as a projection of the first partial derivatives of the surface z(x, y) with respect to x(19)[25] and y(20)[25] with a vector n(21)[25] normal of the surface point.

$$p(x,y) = \frac{\partial z(x,y))}{\partial x} \quad (19)$$

$$q(x,y) = \frac{\partial z(x,y))}{\partial y} \quad (20)$$

$$n(x,y) = \frac{(p,q,-1)}{\sqrt{1 + \|\bigtriangledown z(x,y))\|^2}} \quad (21)$$

if we considered that the direction of light source L is the same as the camera direction V, the irradiance image equation (7) defined as (22), for $\omega_s = 0$ (22) can be define as (8) and solver using (9), for the unified reflectance $\omega_s \neq 0$, needs to apply the Newton–Raphson method (23) [9] using the definition (26), as strategy to accelerate the calculations we applied the fast eikonal solvers based on the numerical method of Godunov-Based [9].

$$I(X,y) = \omega_d \left(\frac{A}{\sqrt{1 + \|\bigtriangledown z(x,y))\|^2}} + \frac{B \bigtriangledown z(x,y)^2}{1 + \|\bigtriangledown z(x,y)\|^2} \right) + \omega_s \left(\frac{1}{\sqrt{1 + \|\bigtriangledown z(x,y)\|^2}} \right)^n \quad (22)$$

$$|\bigtriangledown z(x,y)| = \sqrt{\frac{1}{(T^k)^2} - 1}, \forall (x,y) \varepsilon \Omega, where$$
$$T = \frac{1}{\sqrt{1 + \|\bigtriangledown z(x,y))\|^2}} \quad (23)$$

For solving (23), we must be defined (22) as (24) and its first derivate(25)[9].

$$F(T) = \omega_s T^n - B\omega_d T^2 + A\omega_d T + B\omega_d - I \quad (24)$$

$$F^{'}(T) = n\omega_s T^{n-1} + \omega_d (A - 2BT) \quad (25)$$

$$T^k = T^{k-1} - \frac{F(T^{k-1})}{F^{'}(T^{k-1})} \quad (26)$$

Indoor Visual Semantic SLAM improves VIO and RGBD for narrow space navigation

Andrés Solares Jurado†, Germán Andrés Di Fonzo,† Rafael Pérez†, Hriday Bavle ‡, Miguel Fernandez-Cortizas†, Javier Rodríguez-Vázquez †*, Guillermo Robledo†, Pascual Campoy†
Computer Vision and Aerial Robotics group (CVAR), Universidad Politécnica de Madrid †
University of Luxembourg ‡

ABSTRACT

Self localization in GNSS denied environments is a key requirement for Micro Aerial Vehicle (MAV) applications. Indoor environments have an abundant presence of high-level semantic information that can be exploited to improve the environment understanding, as well as their pose estimation. Given that MAVs cannot carry much weight, they can only be equipped with light sensors, such as RGB-D cameras, and processing units with limited computational resources. In this paper, we tackle the problem of real-time visual semantic SLAM running on-board lightweight aerial robotic platforms by using the OAK-D RGB-D sensor. The proposed algorithm is divided into two parts. In the first part, the robot state is propagated using VO/VIO estimation by using low-level features of the environment. Then, to counterpart the accumulated drift of such low level algorithms, we associate high-level sensed objects with previously mapped landmarks. Thus, the second part of the algorithm corrects the estimation and builds a sparse semantic map of the landmarks extracted from the detections.

1 INTRODUCTION

Our aim is to provide MAV the ability to navigate around narrow constrained spaces during disasters . This kind of vehicles cannot carry a lot of payload, so they can only be equipped with light-weight sensors, such as RGB or RGB-D cameras (OAK-D), and processing units with limited computational resources. To operate in a truly autonomous way, accurate localization and meaningful mapping results are needed, which is indeed a challenging problem, especially regarding robustness.

Simultaneous Localization and Mapping (SLAM) using visual sensors may be feature-based (sparse, semi-dense or dense) or intensity-based. Most semi-dense SLAM techniques, like [1, 2, 3], rely on low-level characteristic features of the environment such as points, lines, and planes. This kind of approaches typically deteriorate in performance in the presence of illumination changes and repetitive patterns. On the other hand, other state of the art SLAM based techniques, such as [4, 5], focus on dense 3D mapping of the environment, hence requiring high-end CPU and GPU hardware to achieve real-time operation, which is a clear limitation on board an aerial robot with low computational capabilities.

Recent improvements in computer vision algorithms have made it possible to achieve object-based detectors running real-time on lower end CPUs or GPUs. Combining such detectors with Visual Odometry (VO)/ Visual Inertial Odometry (VIO) systems depending on low-level features can improve the accuracy of the data associations and provide more robust loop closure without high computational requirements, as shown in [6, 7]. Although adding semantic information to SLAM systems undoubtedly provides additional knowledge, extracting the accurate 3D position of semantic objects is a challenging problem with important implications, since errors in the position estimation can induce errors in the data association and mapping of such semantic objects.

The inaccuracies in estimating the 3D positions of semantic objects are mainly due to two factors: Uneven and complex 3D structures of different instances of semantic object classes. Errors in the semantic object detections, i.e, bounding boxes provided by the object, detectors do not fit accurately around the detected object.

Most indoor scenarios include some static key objects (screens, tables, windows, etc.) that can be used as reference landmarks. Hence, to overcome the above-mentioned limitations and to achieve a robust and lightweight SLAM algorithm, we use a semantic SLAM approach using key objects within the semantic detections.

In this paper, we present a Semantic SLAM algorithm that can be divided into two parts. In the first part, the robot state is propagated using a VO/VIO estimate. Low-level features from the environment are used at this stage for the propagation of the robot state. Due to the inaccuracies in low-level feature detection and matching, as well as to errors and biases in the IMU measurements (for VIO systems), the VO/VIO estimations of the robot state often accumulate errors over time. We address this by associating the high-level detected objects with the previously mapped key objects.

To estimate the position of the key detections, we integrate state of the art detectors on the OAK-D, using its neural

*Email address: javier.rodriguezvazquez@upm.es inference feature. Being able to perform the detection of both cameras allows us to fuse the output information and gather 3D information. Hence, the second part of the algorithm corrects the estimation and builds a sparse semantic map of the key objects extracted from the detections. The created semantic map consists of centroids along with their class labels and type, which may be augmented by new detections of the semantic objects.

2 Related Work

The research community has already given a great interest in visual SLAM based algorithms applied to robotics. In this section, we are going to review some of the latest and most significant visual SLAM related literature.

Salas-Moreno et al. [8] presented a pioneer work in this direction. A real-time semantic SLAM called SLAM++. This type of SLAM was developed for a RGB-D sensor and it extracts estimates of poses from a 3D camera pose track using the ICP algorithm. It then integrates the relative 3D poses estimated from semantic objects in order to jointly optimize all the poses.

Parkhiya et al [9] Gives the semantic SLAM a monocular approach. With the use of a deep network to learn features of 2D objects to later, match it with a 3D CAD model to estimate the relative pose of the semantic object.

McCormac et al. [10] develops an object-level SLAM system using RGB-D cameras, segmenting Truncated Signed Distance Function (TSDF) representations of the object with help of the Mask-RCNN object detector. This detected objects are used to complete the SLAM algorithm: tracking, re-location and loop closure.

Murali et al. [11] presents an approach which integrates semantic information into a visual SLAM system. The semantic information is used for detecting the inliers/outliers of the system in order to achieve robust performance in the presence of dynamic obstacles. A pre-trained deep learning based object detector provides the semantic information of the objects.

Grinvald et al. [12] propose a semantic mapping system based on a pose acquired from a geometric VIO sensor. This method utilizes geometric planar segmentation of a point cloud data and then uses semantic detections for the data association step and to further refine the segmentation.

One of the most recent publications comes from Yang et al. [13]. He proposed a unified SLAM framework including high-level object detection and planes based on monocular information. Other innovative approaches have focused on point-wise semantic labeling for 3D Lidar data within the SLAM framework [14].

3 System Approach

3.1 Sensor integration

The OAK-D camera has been used as the main sensor for this approach. It integrates a color camera, which allows a maximum resolution of 12MP and a maximum frame rate of 60. In addition, it incorporates stereo camera pair, with a maximum resolution of 1MP each, at a maximum frequency of 120 frames per second. It also integrates an inertial measurement unit (IMU). It is capable of providing spatial information using the three cameras in different ways. The first one, making use of Monocular Neural Inference fused with Stereo Depth, the neural network is run on a single camera (left, right or color camera) and fused with disparity depth information. The second one, using stereo neural inference, the neural network is run in parallel on both the left and right stereo cameras to produce 3D position data.

One of the main characteristics of the camera is that it performs neural network inference. Using Intel's OpenVino compiler, pre-trained models can be loaded into it. As mentioned before, the camera obtains three-dimensional information from objects, which allows the fusion of an object detector neural network with it, providing three-dimensional data of the detected objects. The device not only can run neural networks, but also allows its post-processing, which makes it more flexible and interesting for integration with small robots such as drones, which typically lack high computing power.

Regarding calibration, the OAK-D offers information on the intrinsic and extrinsic parameters of each camera. However, the IMU intrinsic parameters calibration has been calculated using the C++ version of Allan Variance Tool package [15], with which the gyroscope and accelerometer white noise and bias instability are calculated. Thanks to this, we can obtain the IMU covariance matrices needed for the VIO algorithm.

3.2 Object detection

Object detection algorithms locate the presence of specific objects in an image with a bounding box. This will allow us later to extract the semantic information along with the spatial coordinates of the detected item. Even though the semantic object detection can be performed using any object detector, it is preferred to be lightweight and fast in this case, since the computing will be run directly on the camera. This is why it makes sense to use OpenVINO [16] in combination with OAK-D, which allows to load deep learning models into a format that can run with high efficiency on Intel hardware.

A pre-trained MobileNetv2SSD [17] neural network has been used, it has been chosen due to its good ratio between effectiveness and speed. After being compiled in OpenVINO, it allows the camera to carry out both its inference and its post-processing, thus obtaining the bounding box of the detected object, together with its probability. This network has been trained in COCO [18] , allowing the detection of 21 different labels corresponding to common objects (vehicles, people, animals and domestic items). When the inference is made on the color camera, the OAK-D provide the depth of the object using the stereo pair. Furthermore, as will be explained later, inference has also been implemented directly on the stereo pair, achieving after post-processing the three-dimensional position of the object.

3.3 *Object position estimation*

In order to improve the estimated pose of the robot computed through odometry, information from the environment needs to be extracted. We seek to detect common static objects in the robot's surroundings, obtaining not only what type of object it is, but also its position. Then, these data can be used in semantic SLAM algorithms in order to achieve our purpose.

In this paper, two different approaches are implemented to address this issue. In the first one, OAK-D high resolution color camera is used. As explained before, this method uses the MobileNet object detector to identify different objects in the RGB images. Once the item is localized, the depth map provided by the device can be used to obtain the center spatial coordinates of the bounding box, which ideally matches with the center of the object. This solution has two major problems. On the one hand, it is not capable of calculating distances less than 0.7 m and, on the other hand, in objects with holes, even if the detection works properly, the center of the bounding box may coincide with one of these holes, causing the depth estimation to fail, obtaining the depth of the background and not of the detected object. In figure 1 an example of a detected object using this configuration can be seen.

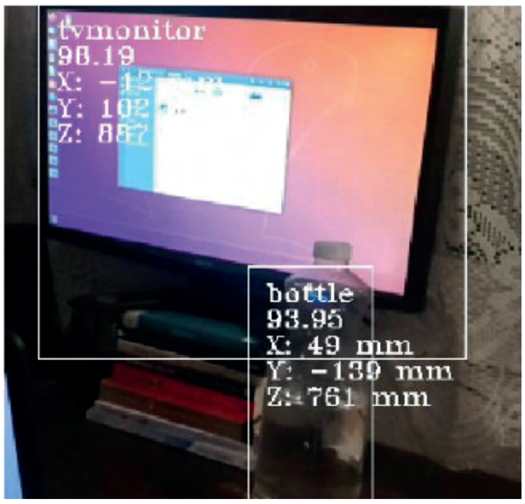


Figure 1: RGB object position estimation approach.

The second approach takes advantage of the stereo pair of the device. First, the images are rectified and, as exposed before, objects are detected in both mono cameras at the same time running two MobileNet neural networks in parallel. Once the bounding boxes of each object have been obtained in each one of the images, the position of the object is computed assuming that both detections are the same, i.e., the detector obtains the same result in both images, and doing a triangulation implementing the stereo pair 3D model. This assumption can be made because the distance between both cameras (baseline) is very small (75 mm), which implies that one image is slightly displaced with respect to the other, so they capture almost completely the same information and, above all, since the cameras are contained within the same plane, from the same perspective.

When same objects have been detected in both images, their bounding boxes have to be matched in order to triangulate the top left and bottom right corners and compute their spatial position, otherwise the process would fail. To avoid false matches, the same objects must appear in both images, so, in a first filter stage, all objects of one class are discarded if they do not appear in equal quantity in the left and right images. Since the images are rectified, determining which bounding box in the left image corresponds to which bounding box in the right image is not a simple task, since it is not possible to directly compare their absolute coordinates with respect to the principal point of the cameras. Depending on the position of the detected object, the bounding boxes may appear in very different coordinates in each image. In order to resolve this problem at a low computational cost, all detections are stored in two vectors (one for each image) and sorted by their relative center's X-coordinate. Thus, when we start comparing all bounding boxes, although several of them could be really similar, if the comparison starts in this order, the chances of matching wrong bounding boxes are hugely decreased.

In a last filter stage, bounding boxes in the first image are compared to the bounding boxes in the second image by the order specified above. To get a match between them, they are filtered by class of object, area, aspect ratio and the center Y-coordinate of the bounding box. Again, even though images are rectified, since the stereo pair is at the same height, the center of the bounding boxes in each image should be very similar (ideally equal), which let us also use this as a reliable filter. By having this amount of variables which can be compared to make a trustworthy match, we can reduce the rigidity of the filters, and so, make the system more robust to inconsistencies in the similarity of the bounding boxes extracted.

Once all bounding boxes are matched correctly, each one of them is processed to obtain the disparity of two diagonal corners by applying the aligned axis pair stereo 3D reconstruction model, which we can be visualized in figure 2.

Disparity $(x_R - x_L)$ is then used to calculate the depth of the corners as shown in equation 1, where f is the focal length and T is the baseline. Once again, in order to make the system more robust, several reductions of the bounding box are carried out to improve depth extraction consistency. Since the final goal is to obtain the position of the object, its centroid is calculated. The bounding box acts as a plane which cuts the object vertically through its center so the Z-coordinate of the centroid is calculated as the average of all corners' depths previously obtained. Once the spatial Z-coordinate of the ob-

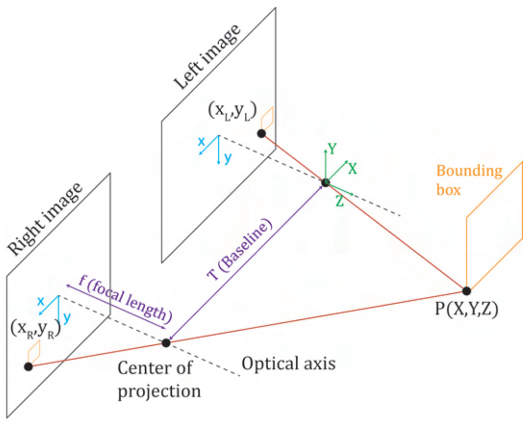


Figure 2: Stereo model triangulation.

ject's centroid is estimated, X and Y spatial coordinates are calculated as shown in equations 2 and 3. These are spatial coordinates which are referred to one of the mono cameras frame. Therefore, the coordinates are transformed to the device frame, making it easier to transform them to the world frame later.

$$Z = \frac{fT}{x_R - x_L} \tag{1}$$

$$X = \frac{Zx_L}{f} \tag{2}$$

$$Y = \frac{Zy_L}{f} \tag{3}$$

In figure 3 an example of detection using this configuration can be observed.

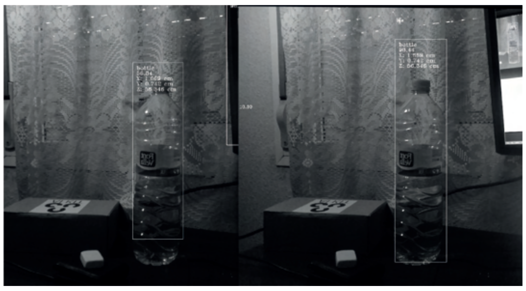


Figure 3: Stereo object position estimation approach.

Algorithms 1 and 2 show the complete process of the both proposed algorithms for object position estimation.

Algorithm 1: Object position estimation using neural inference with RGB images from OAK-D.

```
Result: Spatial coordinates of the centers of the
        detected bounding boxes.
1 Extract RBG image from color camera;
2 Reduce resolution to 300x300 px;
3 Apply MobileNet object detector to RGB image;
4 Compute bounding boxes in depth map;
5 Compute depth of the bounding boxes' centers in
    depth map;
6 Compute X and Y coordinates of the objects from
    their depth;
```

Algorithm 2: Object position estimation using stereo neural inference in OAK-D.

```
Result: Spatial coordinates of the centroid of the
        detected objects.
Extract rectified images of the stereo pair;
Reduce resolution to 300x300 px;
Apply MobileNet object detector to both images;
Scale bounging box to original resolution;
Sort the detections by relative X-coordinate;
for i = 0 to num_right_detections do
    $a_R = area(detections_{right}[i])$;
    $r_R = aspect\_ratio(detections_{right}[i])$;
    for j = 0 to num_left_detections do
        $a_L = area(detections_{left}[j])$;
        $r_L = aspect\_ratio(detections_{left}[j])$;
        if $a_R \approx a_L$ & $r_R \approx r_L$ & $y_R \approx y_L$ then
            Compute $Z_{centroid}$;
            Compute $X_{centroid}$;
            Compute $Y_{centroid}$;
            $detections_{left}[j].erase$;
            break;
        end
    end
end
```

The second algorithm solves the problem of hollow items, if the object is accurately detected in both cameras, since the extracted bounding box represents a plane cutting the center of the object, the estimation will always be consistent, no matter the shape or holes of the object. Also, this approach allows to detect objects with very high resolution at very short distances (below 0.7 m), which is a key factor when navigating in indoor environments and narrow spaces.

3.4 Robot pose estimation

Semantic SLAM algorithms focus on simustaneously locate the robot position in the environment and create a map which represents its surroundings by using semantic information of them. By analyzing and processing information about the environment navigation tasks can be carried out more efficiently. This type of approaches have gained popularity not only because of its great performance but also for the low cost of the sensors used.

Both object position estimation approaches allow semantic SLAM algorithms to correct the estimated pose of the robot given by odometry. Our goal is to use Visual Planar Semantic SLAM algorithm proposed by Hriday Bavle [19], which takes advantage of environments that have abundant presence of high-level semantic information. This algorithm uses RGB-D sensors which provide depth information of the pixels of the image, with this data planar surfaces are extracted from the detected objects. From this calculated planar surfaces the centroid of the object and its normal vector are calculated. Our focus is to simplify the system by removing the planar surface extraction since, as we described before, our object pose estimation algorithms running on OAK-D camera can directly provide the centroid of the selected object along with the semantic information. By erasing the need of detecting objects with planar surfaces the algorithm gains versatility and the ability to detect hollow objects.

3.5 *Graph slam*

As in most navigation algorithms, the pose estimates from the odometry (in this case VO/VIO) which accumulates error, especially in absence of external references or features. We find other source of error in the semantic detections, due to insufficient lighting, occlusions or object detection failure. With these uncertainties the traditional use of filtering techniques such as the Extended Kalman Filter (EFK) SLAM can lead to false results in the estimate of the robot as well as the landmark poses. By using a graph slam approach we ensure all previous robot states are considered. In order to fuse measurements from VO/VIO and semantic extracted information, graph slam based optimization is used.

As mentioned before, our goal is to implement into our detections approaches semantic slam algorithm proposed by Hriday Bavle [19]. The whole algorithm can be divided in 4 main steps.

The first one being the acquisition of odometry using VO/VIO methods, in our case provided by the RealSense T265 camera. The next stages of the algorithm which will be explained in more detail are graph construction, data association and graph optimization.

In regards to graph construction; the robot state vector is defined as $x = [x_r, R_r]$ which is propagated over keyframes $k.X_r$ being the position estimates $x_r = [x, y, z]$ with respect to the world reference W and R_r being the rotation matrix with respect to the world frame W. The starting state of the robot x is assumed to be known.

Odometry provides the 3D pose estimate in the world frame reference W. This estimate of the robot state x_r at time t is added to the graph as a keyframe node K_t. The constraint between adjacent keyframes K_{t-1} and K_t is added in the form of an edge using the pose increment between them $u_r(k)$. The pose increment obtained from the odometry poses at time $t-1$ and time t can be derived as:

$$u_{r_t} = \ominus x_r(k-1) \oplus x_r(k) \tag{4}$$

Each keyframe K_i is added to the factor graph depending on time as well as motion constrains of the robot. Each detected semantic object D_i can be either added as a new landmark node or associated with the currently mapped semantic landmark. Depending on the data association process, a choice is made as to whether the landmark is new or must be mapped to a previously detected landmark.

In the data association step the semantic and spatial information of the detected objects is received as follows: $\mathbf{D}_i = \{d_{z_i}, d_{c_i}, d_{p_i}\}$. Being D_i the detected semantic object i, d_{z_i} the centroid position, d_{c_i} the class label of the object and d_{p_i} the probability of successful detection.

There is no need for the first semantic object to go through the data association step so it is directly added to the graph as a new landmark. It is stored and represented in the following manner:

$$\mathbf{L}_i = \{l_{z_i}, l_{c_i}, l_{\sigma_i}\} \tag{5}$$

where l_{z_i} is the landmark centroid, l_{c_i} the class of the landmark and l_{σ_i} the uncertainty of the estimated position and reliability of the landmark, which is initialized at a high value related to the successful detection probability.

After this process, the next detected objects have to go through the data association phase, where first of all it is checked if their class coincides with any of the already mapped landmarks. If so, the relative spatial position of the centroid is transformed from the camera frame to the world frame using equation 6 to calculate the Mahalonobis distance between the new detected object and the possible already mapped landmarks. If this Mahalanobis distance is greater than a certain threshold it means that the object does not match any landmark and is mapped as a new one. In case this distance is smaller than the threshold the current semantic object is matched with the corresponding landmark.

$$l_{z_i} = x_r \oplus w_{Rc} \cdot d_{z_i} \tag{6}$$

In the last stage, which is the graph optimization, the landmarks positions l_{z_i} and their covariances l_{σ_i} of all the mapped semantic landmarks are optimized and updated. When a previously observed semantic object is seen again and correctly associated to the corresponding landmark, a loop closure is obtained after optimization, which allows the system to correct the robot pose. This graph optimization step can be consulted for more detail in the original algorithm proposal [19].

4 RESULTS

To validate our approach and test the feasibility of the OAK-D camera for SLAM applications the same experiment is carried out with each of the detection approaches. The main objective is to determine whether the camera could be feasible for use in improving odometry estimation in aerial vehicles with low payload capacity. The second objective is to determine which of the two approaches is the most suitable for this task.

The experiment consists of navigating an indoor environment filled with common office objects such as chairs, tables, monitors and bottles in random positions, a picture of the setup can be seen in figure 4. In this setup, an Intel RealSense T265 (with loop-closure disabled) is used in conjunction with the OAK-D as a source of odometry. To compare the odometry data with the semantic slam estimation, we use a motion capture system as ground truth data.

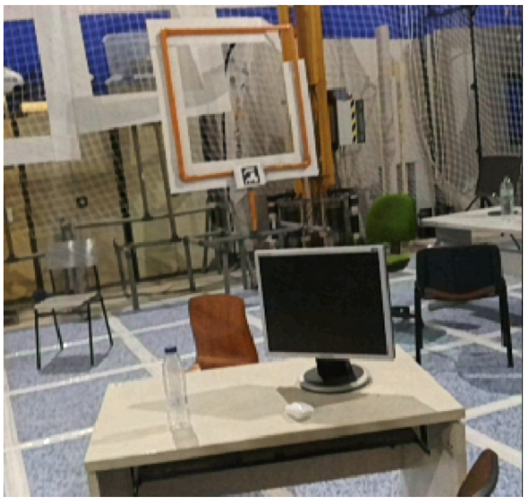


Figure 4: Experiments setup.

To test and compare the results of both approaches, we perform a similar trajectory in the prepared environment using in each case one type of detections, 3 loops are carried out in a repetitive manner. The method proposed in [20] is then used to compare the odometry and estimation trajectories with the ground truth trajectory. Figures 6a and 5a show the performed trajectories and the ground truth data.

In table 1 Average Trajectory Error (ATE) with respect to the ground truth data is presented for each approach and the odometry obtained in the test. Note that the showed values not only refer to the translation estimation but also to the rotation. The measurements included are the Root Mean Square Error (RMSE), the standard deviation and the median. Although the median value is not the most used metric, in this case has been included because of the high RMSE obtained in the rotation estimations due to outliers which can be observed in 5b and 6b.

As shown in table 1, the object position estimation using the RGB image approach is not capable of improving the robot pose estimation, on the contrary, it makes the trajectory differ from the ground truth even more than the VIO path. On the other hand, the second object position detection approach is able to correct the trajectory, reducing the ATE by approximately 0.16m. It demonstrates that this approach outperforms the former approach for semantic SLAM applications in small environments and that it can effectively improve the position estimation of a robot navigating autonomously.

Table 1: Absolute Trajectory Error (ATE) m comparisson between detections approaches and the obtained odometry.

	Trans [m]			**Rot [deg]**		
	RMSE	**Std**	**Median**	**RMSE**	**Std**	**Median**
RGB approach	0.536	0.406	0.244	31.595	28.349	9.011
Odometry	0.504	0.408	0.222	31.262	28.301	8.660
Stereo approach	0.378	0.263	0.200	30.237	27.492	7.769
Odometry	0.540	0.371	0.304	9.860	1.739	9.491

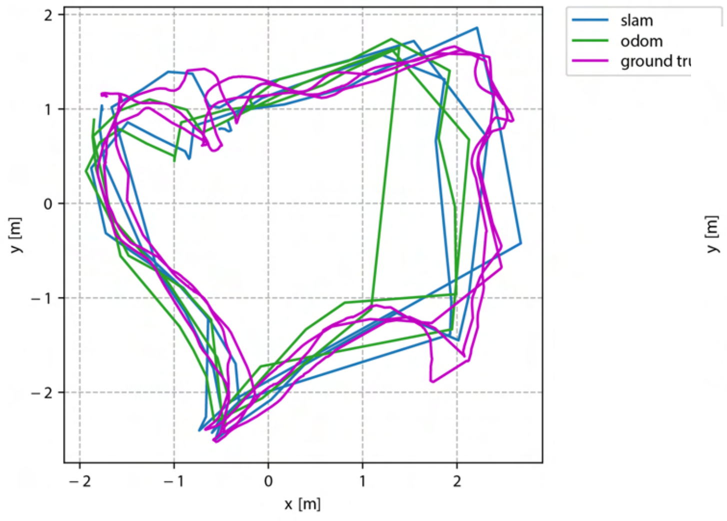
5 CONCLUSIONS

In this paper, we present a semantic SLAM algorithm with the aim of improving low payload aerial vehicles pose estimation. For this purpose, we presented two pose estimation methods for the new OpenCV RGB-D camera, the OAK-D, that extract the centroid of the objects. The first method has shown to be insufficient to accomplish such a task, however, OAK-D is still under development and software updates can make the detections faster and more robust. The second approach, which runs two object detection neural networks in parallel in each of the cameras of the stereo pair, has proven to be a better option for SLAM algorithms, especially when navigating in narrow spaces.

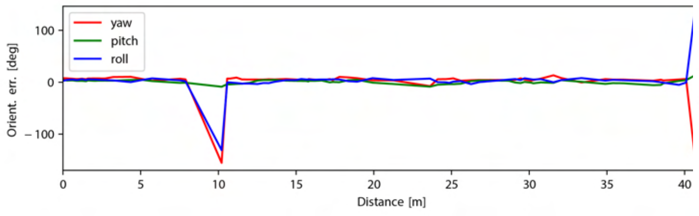
As explained before the first method offers a better precision when detected objects are ”far” (around 1.5-2.5 m), however, the second method works best when objects are close (around 0.2-1.5 m). Another important factor is the process speed of each method, in the first case, since the whole disparity map is calculated, fps drop significantly, reducing it’s effectiveness when on board of an aerial vehicle (which can move quite fast). On the other hand, although the second method runs two neural networks in parallel, it has to process the depth for a much smaller number of points, which translates into a higher computational efficiency, allowing to obtain a larger number of fps.

ACKNOWLEDGMENT

This work has been supported by the project COMCISE RTI2018-100847-B-C21, funded by the Spanish Ministry of Science, Innovation and Universities (MCIU/AEI/FEDER, UE). The work of the first author is supported by the Research Assistant and Laboratory Technicians of the Science of the Science, University and Innovation Department of Madrid Government (partially funded by FEDER/EU) The work of the fifth author is supported by the Re-search Assistant and Laboratory Technicians Program of theScience, University and Innovation Department of MadridGovernment (partially funded by FEDER/EU). The work of the sixth author is supported by Pre- and Post Doctoral Program of the Science,

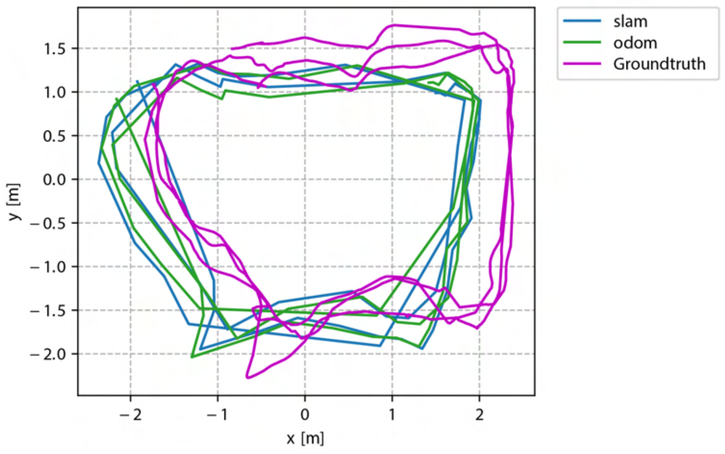


(a)

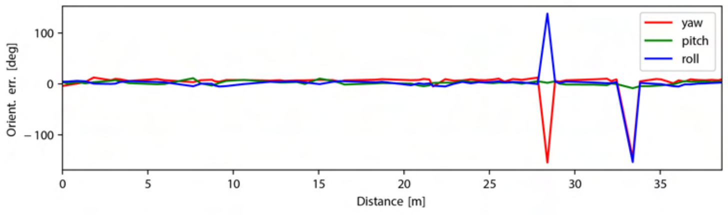


(b)

Figure 5: (a) Ground truth, odometry, and SLAM top view trajectories with stereo detections approach. (b) SLAM rotation error.



(a)



(b)

Figure 6: (a) Ground truth, odometry, and SLAM top view trajectories with RGB detections approach. (b) SLAM rotation error.

University and Innovation Department of Madrid Government (partially funded by FEDER/EU).

References

[1] 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.

[2] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014.

[3] Alexander JB Trevor, John G Rogers, and Henrik I Christensen. Planar surface slam with 3d and 2d sensors. In *2012 IEEE International Conference on Robotics and Automation*, pages 3041–3048. IEEE, 2012.

[4] Thomas Whelan, Stefan Leutenegger, R Salas-Moreno, Ben Glocker, and Andrew Davison. Elasticfusion: Dense slam without a pose graph. Robotics: Science and Systems, 2015.

[5] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *2013 International Conference on 3D Vision-3DV 2013*, pages 1–8. IEEE, 2013.

[6] Nikolay Atanasov, Menglong Zhu, Kostas Daniilidis, and George J Pappas. Semantic localization via the matrix permanent. In *Robotics: Science and Systems*, volume 2, 2014.

[7] Sean L Bowman, Nikolay Atanasov, Kostas Daniilidis, and George J Pappas. Probabilistic data association for semantic slam. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 1722–1729. IEEE, 2017.

[8] J. Kelly Salas-Moreno, Newcombe and A. J. Davison. Slam++: Simultaneous localisation and mapping at the level of objects. *IEEE Conf. Comput. Vis. Pattern Recognit*, pages 1352–1359, Jun. 2013.

[9] R.Khawad P.Parkhiya and K.M.Krishna. Constructing category-specific models for monocular object-slam. pages 1–8, 2018.

[10] A.Davison J.McCormac, R.Clark and S.Leutenegger. Fusion++: Volumetric object-level slam. Sept. 2018.

[11] S.Samarasekera V.Murali, P.Chiu and R.Kumar. Utilizing semantic visual landmarks for precise vehicle navigation. *IEEE 20th Int. Conf. Intell. Transp. Syst. (ITSC)*, pages 1–8, Oct. 2017.

[12] T.Novkovic R.Siegwart M.Grinvald, F.Furrer and J.Nieto. Volumetric instance-aware semantic mapping and 3d object discovery. *IEEE Robot. Autom. Lett*, pages 3037–3044, Jul. 2019.

[13] S. Yang and S. Scherer. Monocular object and plane slam in structured environments. *IEEE Robot. Autom. Lett*, pages 3145–3152, Oct. 2019.

[14] J.Behley X. Chen, A.Milioto and C.Stachniss. Suma++: Efficient lidar-based semantic slam. *IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, page 4530–4537, Nov. 2019.

[15] Oliver J. Woodman. An introduction to inertial navigation. Technical Report UCAM-CL-TR-696, University of Cambridge, Computer Laboratory, August 2007.

[16] Alexander Demidovskij, Yury Gorbachev, Mikhail Fedorov, Iliya Slavutin, Artyom Tugarev, Marat Fatekhov, and Yaroslav Tarkan. Openvino deep learning workbench: Comprehensive analysis and tuning of neural networks inference. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 783–787, 2019.

[17] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. pages 4510–4520, 06 2018.

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[19] Hriday Bavle, Paloma De La Puente, Jonathan P. How, and Pascual Campoy. Vps-slam: Visual planar semantic slam for aerial robotic systems. *IEEE Access*, 8:60704–60718, 2020.

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

Decentralized Trajectory Generation Technique for Multiple Unmanned Multicopter Systems in Cluttered Environments

Xinyi Wang*, Lele Xi, Yizhou Chen, Shupeng Lai, Feng Lin, and Ben M. Chen

ABSTRACT

Challenges in motion planning for multiple quadrotors in complex environments lie in overall flight efficiency and the avoidance of obstacles, deadlock, and collisions among themselves. In this paper, we present a model predictive control (MPC) based gradient-free approach for multiple quadrotors to achieve distributed and asynchronous cooperative motion planning in cluttered environments with the consideration of time consumption. First, the motion primitives of each quadrotor are formulated as the boundary state constrained primitives (BSCPs) which are constructed with jerk limited trajectory (JLT) generation method, a boundary value problem (BVP) solver, to obtain time-optimal trajectories. They are then approximated with a neural network (NN), pre-trained using this solver to reduce the computational burden. Finally, the reference trajectories are generated using the same BVP solver. Our simulation and experimental results demonstrate the superior performance of the proposed method.

1 INTRODUCTION

Trajectory planning for multiple quadrotors is key to execute missions in cluttered environments. In particular, multi-quadrotor tasks are especially challenging due to many decision-making agents sharing the same space. In such settings, the planning algorithms must compute collision-free and goal-oriented trajectories taking into account of the neighboring agents and environment. Furthermore, the requirements of excellent flexibility, flight efficiency and speed for multiple quadrotors system make high demands on planning methods, which should have good computational efficiency and high flight performance, as well as minimizing execution time of trajectory for actual implementations.

A wide variety of techniques exist to tackle the multi-quadrotor trajectory generation problem. MPC-based methods have been proven effective for the motion planning of autonomous vehicles in complex environments. The distributed model predictive control (DMPC) approach [1] is developed due to its abilities to handle constraints and achieve good performance for task coordination (see e.g., [2, 3]). The result in Parys and Pipeleers [4] shows that parallel computing can also be combined with this method to reduce the run time when quadrotors are updating their predicted states. In terms of tracking and formation problems, Wang and Ding [5] presented a synchronous DMPC scheme using the estimated information of other quadrotors to avoid collisions. However, with the increasing of environments and task complexity, it is hard to handle state and input constraints well and guarantee trajectory feasibility. Real-time trajectory generation is required for quick adaptation in complex environments, but it remains challenging to implement for robot swarms. Most obstacle avoidance techniques for trajectory generation are either centralized or sub-optimal (see e.g., [6, 7]) usually with high trajectory execution time.

Moreover, there are many methods formulating trajectory generation as a non-linear optimization problem that takes smoothness and safety into account. Motion primitives (MPs) based local planning methods for mobile robots are also frequently applied to abstract the continuous state space [8]. The MPs along the whole trajectory are sampled on the vehicle's boundary state constraints (i.e. jerk), and then generate the actual motion by solving a boundary value problem (BVP) [9]. In addition, considering the time efficiency, the jerk limited trajectory (JLT) has been proven well suited for quadrotors since it could satisfy the dynamic limits well [10]. It can handle arbitrary initial states for the position, velocity, and acceleration, and resulting in a smooth and time-optimal trajectory from the current state to the next target state.

Inspired by the existing researches, in this paper, we develop a novel trajectory generation method by solving a non-convex optimization problem to achieve distributed cooperative motion planning with considering deadlock and flight efficiency for multiple quadrotors. The main contribution of our work is to propose a decentralized and meta-heuristic, a gradient-free motion planning framework based on MPC which allows for fast trajectory generation for multiple quadrotors in cluttered environments. Unlike gradient-based method, the replanning time is more uniform and thus can improve the success rate. More specifically, we use JLT approximated with NN to reduce the time consumption of trajectories and rapidly generate trajectories with a less computational burden. Simulation and experimental results

*Email address(es): xywangmae@link.cuhk.edu.hk
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong

show that for different environments and boundary states, our method can generate dynamically feasible trajectories for multiple quadrotors and guide them to achieve their goals in an obstacle-dense environment without deadlocks. The proposed framework is tested using actual quadrotors, and the flight experiments are carried out in cluttered environments with static and dynamic obstacles to verify the planning performance.

The rest of the paper is organized as follows. We present in Section 2 some preliminary knowledge and the problem formulation, and in Section 3 we propose our MPC-based trajectory generation framework and analyze the detailed techniques of the trajectory generation problem for multiple quadrotors. The experimental results and analysis are then given in Section 4 to validate the proposed technique. Finally, we draw some concluding remarks in Section 5.

2 Problem Formulation

We separate the flight control problem into an outer loop and an inner loop. For the quadrotor, the outer loop stabilizes transnational variables and generates a reference signal fed to the inner loop. In our paper, we use the nominal model in [11] to generate the outer loop trajectory, which acts as the reference to the inner loop. We consider the trajectory planning problem for a multi-agent quadrotor system with K quadrotors in the 3D space $\mathcal{X} \subset \mathbb{R}^3$. Every quadrotor k with $k \in K$ is assumed to obey the same dynamic limits and its dynamics is differentially flat as adopted in Mellinger and Kumar [12]. The quadrotor dynamics has inputs $\sigma_k = [p_k, \psi_k]^{\mathrm{T}}$, where $p_k \in \mathbb{R}^3$ is the position of the mass center of quadrotor in the world frame, and ψ_k is the yaw angle. This allows us to plan the trajectory of quadrotor k independently using a triple integrator with its position p_k, velocity v_k, acceleration a_k, and jerk j_k, respectively. Let $\mathrm{x}_k = [p_k, v_k, a_k]^{\mathrm{T}}$ be the state variable with invariant constraints $v_k \in [v_{\min}, v_{\max}]$, $a_k \in [a_{\min}, a_{\max}]$ and $j_k \in [j_{\min}, j_{\max}]$ which might be different for the horizontal and vertical axes. $u_k = j_k$ is defined as the control input and Δt is the sampling interval. Thus, the discrete-time linear model can be defined as follows:

$$\mathrm{x}_k[n+1] = A_k \mathrm{x}_k[n] + b_k u_k[n], \tag{1}$$

where

$$A_k = \begin{bmatrix} 1 & \Delta t & \Delta t^2/2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}, \quad b_k = \begin{bmatrix} \Delta t^3/6 \\ \Delta t^2/2 \\ \Delta t \end{bmatrix}. \tag{2}$$

We aim to generate a continuous, smooth, collision free and kinodynamical feasible trajectory $\mathcal{F}_k$ of each quadrotor k. The state vector of each quadrotor is denoted as $\mathrm{x}_k(t) \in \mathcal{X}$ at runtime t. The static obstacles are represented using an occupancy grid map and the dynamic obstacles are defined as a set of convex moving obstacles. It is assumed that we can predict the trajectory of each quadrotor and penalize the relative distance of them to avoid crashes. The free configuration space for a single quadrotor is defined as $\mathcal{X}_{\text{free}}$. All trajectories are considered collision-free if, for each quadrotor, there are no collisions between the quadrotor and environment:

Considering the control objective, we adopt in the following an optimization cost function for generating control input sequence u_k, which has three main components: input variation penalty, desired set position penalty and collision-free penalty:

$$\begin{aligned} \min \ J_k = \int_{t_0}^{t_0+T} & \{w_1 u_k^2(t) + w_2 ||\mathrm{x}_k(t) - g_k||^2 + \\ & w_3 e^{-||d_o^k(t)||} + w_4 ||d_s^k||^2\} dt \\ \text{s.t.} \quad & \dot{\mathrm{x}}_k(t) = f(\mathrm{x}_k(t), u_k) \\ & g(\mathrm{x}_k(t), u_k) < 0 \\ & h(\mathrm{x}_k(t), u_k) = 0 \end{aligned} \tag{3}$$

where w_1, w_2, w_3, w_4 are used to trade off these four penalty. For the constraints of trajectory generation problem, $f(\cdot)$ is the nominal model of the quadrotor, which is a continuous version defined by Eqs. 1 and 2. Inequality constraint $g(\cdot)$ indicates the limit interval of velocity, acceleration and jerk, respectively, for three axes. The boundary state constraint $h(\cdot)$ regulates the triple integrator from an initial state configuration s_k to an assigned goal state g_k. Here we note that the first term is to penalize the square of jerk is to generate smooth trajectories. The second term is to minimize the deviation from the current state to the desired goal state. The third term is flight safety cost of moving obstacles to handle the collision of potential threats. Differing from the geometrical constraints, we can obtain the relative closest distance $d_o^k(t)$ at time t between the predictive trajectories of quadrotor and moving obstacles. Besides, for flight safety cost of static and non-convex obstacles, we calculate a potential function over a Euclidean Distance Transform (EDT) map giving the quadrotor global vision to avoid suffering from deadlock point. The cost to the goal point of every grid for each quadrotor d_s^k is a value stored on this map. As shown in Figure 1, this collision cost pushes the quadrotor away from static and dynamic obstacles to satisfy the collision-free requirements

By solving the optimization problem, a locally optimal sequence of commands during each predictive horizon can be attained to drive each quadrotor to reach its desired destination while avoiding collisions.

3 Framework of Multi-quadrotor Trajectory Generation

In this section, a synchronous and decentralized non-convex optimization procedure based on MPC is proposed to achieve distributed cooperative motion planning with considering deadlock and flight efficiency. The overall structure of our proposed method is depicted in Figure 2, which consists of offline training NN process and MPC-based optimization. The proposed framework consists of the following two stages:

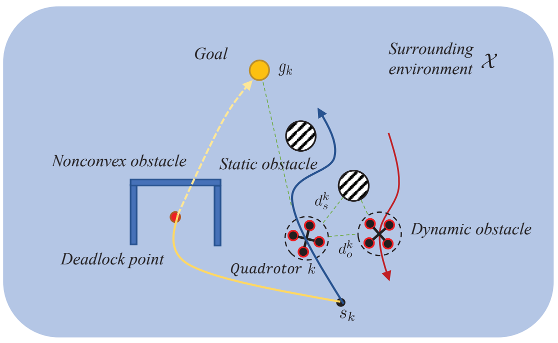


Figure 1: The flight safety of quadrotor k.

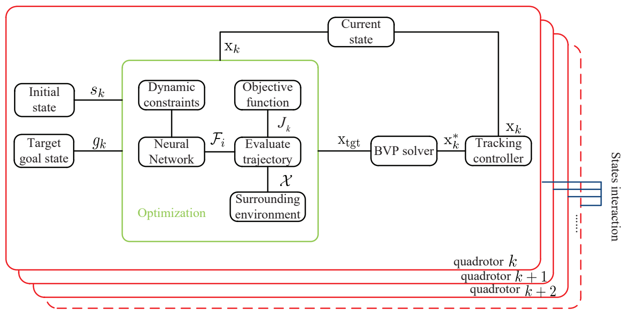


Figure 2: The overall structure of the proposed method for multi-quadrotor motion planning.

1. Training NN stage: We construct the BSCPs, a trajectory library, designed through JLT generation method, a BVP solver, with the given start and goal states. Thus, an NN can be pre-trained to approximate the generated BSCPs to save the time consumption.

2. MPC stage: Given the objective function J_k, we can fast evaluate all candidate end states of quadrotors which are approximated by means of NN with consideration of the surrounding environments $\mathcal{X}$. After this optimization process, we can obtain the end-state constraints x_{tgt} among these candidates to minimize the cost function J_k given in Equation 3 at each step. In addition, we also add guidance map to help the quadrotors to avoid deadlock. After obtaining x_{tgt}, a series of reference state x_k^* can be generated through JLT and only the first few states will be given to the low level dynamic controller. Thus, recomputing the system state and a current state x_k can be returned for the next planning horizon.

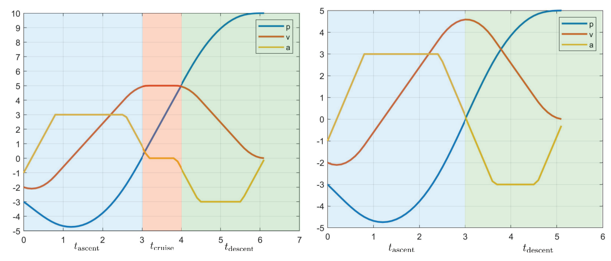
3.1 Construction of BSCPs using JLT

In this subsection, we present detailed procedures for the offline training stage. To prepare for training data, we use a BVP solver associated with the JLT generation method to create a certain range of trajectories of quadrotors in the state space, which are formulated as BSCPs.

The JLT method can solve the trajectory optimization problem and obtain an analytical solution under the condition of limited computing power. Specifically, it can compute the time-optimal trajectory to a set goal $x(t_{\text{end}}) = [p_{\text{ref}}, v_{\text{ref}}, a_{\text{ref}}]$ for the quadrotor system in Eqs. 1 and 2 with arbitrary initial state values $x(0) = [p_0, v_0, a_0]$ and with physical limits $v_{\max}$, $a_{\max}$, $j_{\max}$. The jerk profile is given as $j \in \{j_{\min}, j_{\max}, 0\}$ and the state variables $x(t)$ can be obtained by integrating the jerk profile $j(t)$ at time index t. The time-optimal JLT generation problem can then be formulated as

$$\begin{aligned}
\min \quad & t_{\text{end}} \\
\text{s.t.} \quad & p(0) = p_0, \quad p(t_{\text{end}}) = p_{\text{ref}}, \quad \dot{p}(t) = v(t) \\
& v(0) = v_0, \quad v(t_{\text{end}}) = v_{\text{ref}}, \quad \dot{v}(t) = a(t) \\
& a(0) = a_0, \quad a(t_{\text{end}}) = a_{\text{ref}}, \quad \dot{a}(t) = j(t) \\
& -v_{\max} \le v(t) \le v_{\max} \\
& -a_{\max} \le a(t) \le a_{\max} \\
& -j_{\max} \le j(t) \le j_{\max},
\end{aligned} \tag{4}$$

The algorithm will calculate the covered area of the velocity that equals the desired change in position. The time distribution can be divided into the ascent phase, descent and cruise phases, respectively. The problem is to determine the velocity profile and switching times subject to the state and input constraints. First, it is to accelerate velocity to the maximum with constant jerk $-j_{\max}$, and then to decelerate the speed to zero with $j_{\max}$ to determine whether the end position exceeds the desired position. If the end position is undershooting compared with desired position, i.e., the triangular case in Figure 3, then we use bisection searching algorithm given in [13] to calculate the acceleration time t_{ascent} and deceleration time t_{descent}. Otherwise, it is corresponding to the trapezoidal case in Figure 3, we can also use the position area to determine the acceleration time t_{ascent}, cruise time t_{cruise} and deceleration time t_{descent}.



(a) Trapezoidal case (b) Triangular case

Figure 3: Three types of velocity profile.

After generating the BSCPs, we then implement the trajectory approximation and evaluation system using a pre-trained NN in [9], which is called S_{NN} to save computational time for real flight implementations.

3.2 Construction of Guided Map

At the MPC stage, we use a guidance map to predict information of future states navigating the quadrotor to choose an optimal action. We follow the work of Lau et al. [14] to use an EDT module to represent a surrounding environment. The closest distance of the static obstacles and index for every cell can be efficiently obtained through this map. In this subsection, we propose a method to calculate a potential function from a set of goals to the current positions of the quadrotor combined with an EDT map. We use the Dijkstra algorithm to calculate the cost value c_g to the goal point for each cell p_i, which is the definition of the potential function over the state space. Index i is associated with the revolution of the map. The potential function has the ability to push the quadrotor away from the obstacles and guide it to the goal. Besides, it is quite suitable for the quadrotor, a holonomic system, as this type of robot can freely move in any direction. After constructing the EDT map and potential function, we can combine them together, so the cost value of a cell $\mathcal{M}(\mathrm{p}_i)$ consisting of a cost-to-goal value c_g and a safety value c_s with consideration of the closest distance to obstacles can be calculated as:

$$\mathcal{M}(\mathrm{p}_i) = c_s(\mathrm{p}_i) + \alpha_g c_g(\mathrm{p}_i) \tag{5}$$

where α_g is a weight factor that trades off the relative importance of the cost-to-go value, seen Figure 4. It can guide the local planner to choose appropriate actions u^* to the lowest-cost path of a map at every point during execution, which considers both environments and goal information, thus avoids falling into local minimal, seen Equation 6.

$$u^* = \arg\min_{u} \mathcal{M}(\mathrm{p}_i) \tag{6}$$

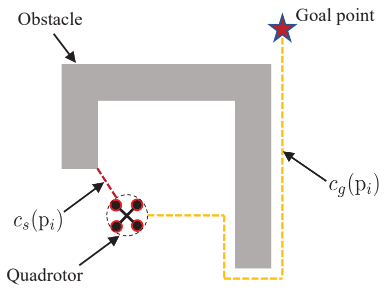


Figure 4: The illustration of deadloack avoidance.

3.3 Trajectory Generation Strategy

We combine the BSCPs with a gradient-free technique, i.e., the particle swarm optimization (PSO), for trajectory planning. This method is capable of finding a feasible solution for each quadrotor in the team to either non-convex or non-continuous problem without reaching the maximum iterations. The other advantage of such meta-heuristic algorithms is that the per iteration time is quite uniform thus avoids failure greatly. The detailed optimization process is shown in Algorithm 1. First, calculate the map $\mathcal{M}_k$ using the method in the previous subsection (Line 2). Then, perform random initialization of particles in the search space (Line 3). Each particle represents a terminal state constraint for one planning horizon. Next, update the velocity δ_i and position $\mathrm{x}_{\mathrm{tgt}_i}$ of particles according to the equation in this algorithm (Lines 5–6), where ω_1 and ω_2 are acceleration constants and $\mathrm{x}^*_{\mathrm{tgt}_i}$ and $\mathrm{x}^g_{\mathrm{tgt}}$ represent the best position experienced by the particle i and the best position experienced by the particles group for the previous interactions. After updating the state of particles, the trajectory $\mathcal{F}_i$ can be approximated using S_{NN} for fast evaluation. The objective function will consider the information of trajectory and the surrounding environment to get the cost value $cost_i$ (Lines 7–8). We can then update the local best and global best position of particles (Lines 9–10) to get the optimal end state, where $cost^*_i$ represents local best cost value and $cost^g$ is the corresponding global best value. The feasible solution $\mathrm{x}_{\mathrm{tgt}}$ will be found after multiple interactions, which is used in a BVP solver to obtain a $\mathcal{F}_k$ of each quadrotor (Lines 12–14).

Algorithm 1 Kinodynamic MPC planner using Particle Swarm Optimization

Input: The surrounding environment $\mathcal{X}$, each quadrotor's state x_k, and corresponding target's goal state g_k;
Output: Trajectory of each quadrotor $\mathcal{F}_k$;

```
1:  for Each planning horizon do
2:      $\mathcal{M}_k$ = getGuidedMap($g_k$, $\mathcal{X}$);
3:      ParticleInitialization();
4:      for Each $\mathrm{x}_{\mathrm{tgt}_i}$, $i = 1, 2, \ldots$ do
5:          $\delta_i = \delta_i + \omega_1 * (\mathrm{x}^*_{\mathrm{tgt}_i} - \mathrm{x}_{\mathrm{tgt}_i}) + \omega_2 * (\mathrm{x}^g_{\mathrm{tgt}} - \mathrm{x}_{\mathrm{tgt}_i})$
6:          $\mathrm{x}_{\mathrm{tgt}_i} = \mathrm{x}_{\mathrm{tgt}_i} + \delta_i$
7:          $\mathcal{F}_i$ = approxTrajectory($\mathrm{x}_k, \mathrm{x}_{\mathrm{tgt}_i}, S_{\mathrm{NN}}$);
8:          $cost_i = J_k(\mathcal{F}_i, \mathcal{M}_k)$
9:          Update local best $cost^*_i$, $\mathrm{x}^*_{\mathrm{tgt}_i}$
10:         Update global best $cost^g$, $\mathrm{x}^g_{\mathrm{tgt}}$
11:     end for
12:     $\mathrm{x}_{\mathrm{tgt}} = \mathrm{x}^g_{\mathrm{tgt}}$
13:     $\mathcal{F}_k$ =JLTTrajGenerotion($\mathrm{x}_k, \mathrm{x}_{\mathrm{tgt}}$);
14:     Return $\mathcal{F}_k$;
15: end for
```

4 Simulation and Experimental Results

In this section, we present our simulation and experimental results to demonstrate the feasibility and robustness of the proposed framework of motion planning for multi-quadrotors in dynamic cluttered environments. We evaluate different settings for the parameters in our planner and compare its performance for different tasks against the state-of-the-art motion planners in the literature.

4.1 Implementation Details

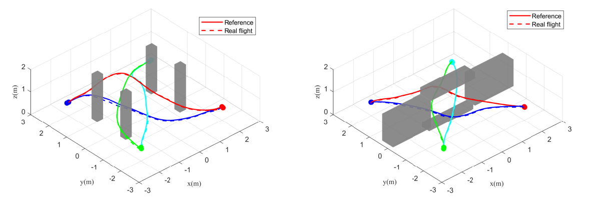
We use Crazyflie, a small, versatile quadcopter, as the experimental platform to verify the proposed method for motion planning of multi-quadrotors. The real flight experiments are carried out in an indoor VICON environment. The VICON system provides localization and obstacles sensing for each quadrotor. The following are three scenarios conducted through simulations and experiments:

- Scenario 1: Each quadrotor must reach its destination while avoiding obstacles in cluttered environments containing several pillars.
- Scenario 2: Each quadrotor must reach its destination while passing through a bridge and guaranteeing the flight safety simultaneously in the vertical axis.
- Scenario 3: Each quadrotor must reach its destination while avoiding deadlock in cluttered environments containing non-convex obstacles.

4.2 Simulation and Flight experiment

To obey the limited physical performance, the kinodynamic feasibility constraints are specified on the velocity, acceleration, and jerk with $v = [2.0, 2.0, 1.0]$, $a = [3.0, 3.0, 2.0]$, and $j = [5.0, 5.0, 5.0]$, which consist of the maximum horizontal limit, minimum and maximum vertical limits, respectively. The environment is represented as a 3D grid map and is then processed to the guided map in Section 3. For the PSO algorithm, 40 candidates of population are iterated over 20 times to find the best candidate solution. The model predictive planning along the process is executed at 5 Hz with a prediction horizon of 2 s.

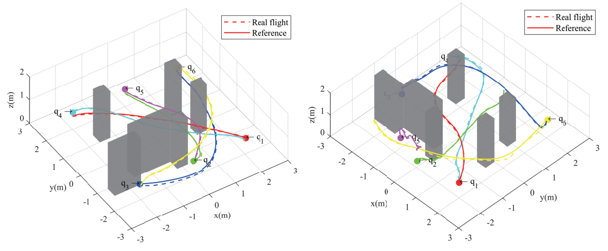
Figure 5 shows cases that the four quadrotors start from their initial positions to their antipodals while avoiding collision with other quadrotors and static obstacles. All the quadrotors avoid the obstacles and converge to their antipodal positions in about 4.8 s. The maximum speed reached is 1.99 m/s with an average speed of 0.93 m/s.



(a) An environment with several pillars (b) An environment with bridge openings

Figure 5: Quadrotor trajectories during antipodal position swapping. In both (a) and (b), the solid circles denote four quadrotors and the solid curves denotes their trajectories, respectively.

As it can be clearly seen in Figure 6, the six quadrotors q_i, $i = 1, 2, \cdots, 6$, carry out the antipodal position swapping in the environment containing several pillars and non-convex obstacles which may result in local minima in optimizing the objective function. In Figure 6(a), Some exciting results can be found that quadrotors, benefiting from the potential function, reach their goal points, and avoid the deadlock simultaneously. As a contrast, in Figure 6(b), the quadrotor q_5 falls into local minima, and it cannot reach its goal successfully without the potential function. Both scenarios show that multi-quadrotors can achieve tasks successfully. Detailed flight experiments can be found in the video clips at `https://youtu.be/QgHfa2dgvv8`.



(a) With potential function (b) Without potential function

Figure 6: The 3D navigation of multi-quadrotors in cluttered environments with non-convex obstacles.

4.3 Comparison with Other Method

We compare our method with methods in Augugliaro et al. [15], Park et al. [7] and Lai et al. [9] on a 10m×10m×2.5m obstacle-free environment using 8 quadrotors and the same boundary states. Detailed results are shown in Table 1.

1) Safety Ratio: The safety ratio is defined as $\min d_o^k / r_c$ for all k, where r_c is an expanding radius of a quadrotor. We have tested the flight safety 20 times and obtain the minimum safety ratio of 1.16, which is still over 1, avoiding collisions with obstacles. Obviously, the SCP-based method sacrifices the safety of a real flight system.

2) Total Flight Time: our method performs better in terms of flight efficiency than the algorithm in Park et al. [7] work as it uses JLT to get the time-optimal trajectory, thus having a shorter total flight time.

3) Time Per Iteration: For each planning horizon, the average time of our work is 0.095s for each quadrotor, satisfying real flight requirements. Compared with the DP method in Lai et al. [9], this method has better performance in terms of reducing the computational burden. Besides, as the number of quadrotors increases, the computational time will only fluctuate slightly due to the quadrotor system is distributed.

Table 1: Summary of flight performance.

Method	Safety Ratio	Time Per Iteration	Total Flight Time
Our Method	1.19	0.095	56.4
DP Method [9]	1.16	0.132	58.4
SCP ($h = 0.34$ s) [15]	0.92	16.2	—
Algorithm in [7]	1.01	—	75.61

5 Conclusion

In this paper, we have presented a decentralized MPC-based trajectory planning method of multi-quadrotors in cluttered environments. The motion primitives along the flight trajectory are generated using the jerk limited method with given initial and goal states and approximated by a pre-trained NN during the optimization process. Furthermore, a gradient-free algorithm, differential evolution, has been applied to find the best solution to meet the requirements of flight safety, efficiency, and kinodynamic feasibility. The proposed method has been tested with simulations and real flight experiments on multi-quadrotors. Experimental results have demonstrated that the proposed method has excellent performance for motion planning of multi-quadrotors in dynamic cluttered environments.

Acknowledgements

This work is supported in part by the Research Grants Council of Hong Kong SAR (Grant No: 14209020), and in part by the Peng Cheng Laboratory.

References

[1] M. Rufli J. Alonso-Mora, A. Breitenmoser and et al. Optimal reciprocal collision avoidance for multiple non-holonomic robots. *Distributed Autonomous Robotic Systems*, pages 203–216, 2013.

[2] H. Rezaee and F. Abdollahi. A decentralized cooperative control scheme with obstacle avoidance for a team of mobile robots. *IEEE Transactions on Industrial Electronics*, 61:347–354, 2014.

[3] B. H. Krogh E. Camponogara, D. Jia and S. Talukdar. Distributed model predictive control. *IEEE Control Systems Magazine*, 22:45–52, 2002.

[4] R. V. Parys and G. Pipeleers. Distributed model predictive formation control with inter-vehicle collision avoidance. In *2017 11th Asian Control Conference*, 2017.

[5] P. Wang and B. Ding. A synthesis approach of distributed model predictive control for homogeneous multi-agent system with collision avoidance. *International Journal of Control*, 87:52–63, 2014.

[6] W. Hönig M. Debord and N. Ayanian. Trajectory planning for heterogeneous robot teams. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018.

[7] I. Jang J. Park, J. Kim and H. J. Kim. Efficient multi-agent trajectory planning with feasibility guarantee using relative bernstein polynomial. In *2020 IEEE International Conference on Robotics and Automation*, 2020.

[8] T.H. Lee M. Lan, S. Lai and B. M. Chen. A survey of motion and task planning techniques for unmanned multicopter systems. *Unmanned Systems*, 9:165–198, 2021.

[9] M. Lan S. Lai and B. M. Chen. Model predictive local motion planning with boundary state constrained primitives. *IEEE Robotics and Automation Letters*, 4:3577–3584, 2019.

[10] E. Weitnauer R. Haschke and H. Ritter. On-line planning of time-optimal, jerk-limited trajectories. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008.

[11] F. Gao, L. Wang, B. Zhou, X. Zhou, J. Pan, and S. Shen. Teach-repeat-replan: a complete and robust system for aggressive flight in complex environments. *IEEE Transactions on Robotics*, 36:1526–1545, 2020.

[12] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, 2011.

[13] S. Lai, M. Lan, Y. Li, and B. M. Chen. Safe navigation of quadrotors with jerk limited trajectory. *Frontiers Inf Technol Electronic Eng*, 20:107–119, 2019.

[14] C. Sprunk B. Lau and W. Burgard. Efficient grid-based spatial representations for robot navigation in dynamic environments. *Robotics and Autonomous Systems*, 21:1116–1130, 2013.

[15] A. P. Schoellig F. Augugliaro and R. D'Andrea. Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

Texture Classification for Object Detection in Aerial Navigation using Transfer Learning and Wavelet-based Features

J.M. Fortuna-Cervantes*[1], M.T. Ramírez-Torres[2], M. Mejía-Carlos[1], J. Martínez-Carranza[3,4] and J.S. Murguía-Ibarra[1]
[1]Universidad Autónoma de San Luis Potosí, Facultad de Ciencias-IICO, San Luis Potosí, México
[2]Universidad Autónoma de San Luis Potosí, Coordinación Académica Región Altiplano Oeste, San Luis Potosí, México
[3]Instituto Nacional de Astrofísica, Óptica, y Electrónica, Puebla, México
[4]University of Bristol, Bristol, UK

ABSTRACT

The use of Micro Aerial Vehicles (MAVs) has increased in engineering and civil applications to explore environments without previous information. In particular, in autonomous navigation, a fundamental part is that of detecting and locating targets of our interest. For this reason, computer vision has become an essential analysis tool. In this work, we focus on object classification in aerial navigation tasks, where texture is involved as a physical property of the object. We present a classification model using transfer learning and wavelet-based features as an additional feature extraction method. This model is trained with the Describable Textures Database (DTD), and a performance of 53% accuracy is obtained. Moreover, the images obtained from the environment show the generalization of learning for some database classes. Transfer learning fusion with wavelet analysis is recommended for small data sets of images with textures due to the limitation of learning about spectral information lost in conventional Convolutional Neural Networks (CNNs).

1 INTRODUCTION

Texture analysis is a traditional problem in computer vision because it involves obtaining information that describes the image content. In the robotics area, object detection is a problem for robots that perform tasks in real scenarios and in real-time, given the lighting conditions, indeterminate orientations, object identity, shape, color and texture. Furthermore, the information may differ in outdoor and indoor environments, which varies the target information [1]. Providing the resources to the robot by integrating sensors can improve object detection.

Micro Aerial Vehicles (MAVs) have been used in different environments due to their easy control and implementation of algorithms through computer vision. In tasks of the classification, detection and localization of the target. There are different vision methods in the area, such as optical flow, segmentation, edge detector, morphological operations and feature extractor for different tasks. These methods have been combined to improve detection performance while the MAV executes its aerial navigation (recognition) or autonomous flight. However, using these methods can be computationally expensive to perform real-time detection, affecting the overall system performance.

In image processing, texture can be defined from neighboring pixels and intensity distribution over the image [2]. Besides, there are some classification methods for texture analysis such as statistical, geometric, model and spectral. If we focus on spectral methods, these methods describe the texture in the frequency domain. They are based on the decomposition of a signal in terms of basis functions. Furthermore, they use the expansion coefficients as elements of the feature vector.

Deep learning has become a helpful tool for image classification, object detection, and segmentation. Especially if we talk about convolutional neural networks, these achieve learning multiple features to recognize targets without reference to their position, indeterminate orientations, scale, and target rotation. VGG16, VGG19, AlexNet, SSD and YOLO are architectures that have good performance in image classification and object detection tasks.

Therefore, we decide to merge these methodologies (deep learning and wavelet features) as a solution for texture classification. The objective is that the MAV performs the aerial navigation (inside the virtual environment) for the classification system to recognize the object, see figure 1. This work focuses on preview information (in data collected by MAV) and structural recognition of the object (with a particular texture) within a region of interest in the image plane.

The implementation of our system is developed with the fusion of two approaches. The first is in the spatial domain, using transfer learning. We take as a baseline the VGG16 architecture with the features of the ImageNet database. The second approach focuses on the spectral domain, applying the

*Email address(es): juan.manuel.fortuna@hotmail.com

Haar wavelet transform in two dimensions to obtain features at different scales [3]. The VGG16 network has been selected for its fast performance and implementation with transfer learning and adaptability with wavelet analysis. Internally, the system is divided into two stages: the first corresponds to feature extraction and the second one to the classification stage. We used the Describable Texture Database (DTD) to train our model, which contains 47 texture classes, with 120 images per class. On average, we have tested with some textures for the classification task in the virtual environment, and the prediction can be performed correctly, with an average processing speed of 2 fps.

The rest of the paper is organized as follows in Section 2 related work is shown, Section 3 introduces the methodology to approach the texture classification problem. Whereas Section 4 shows the results with the DTD dataset and the experimental part to test our model. Finally, Section 5 presents the conclusions.

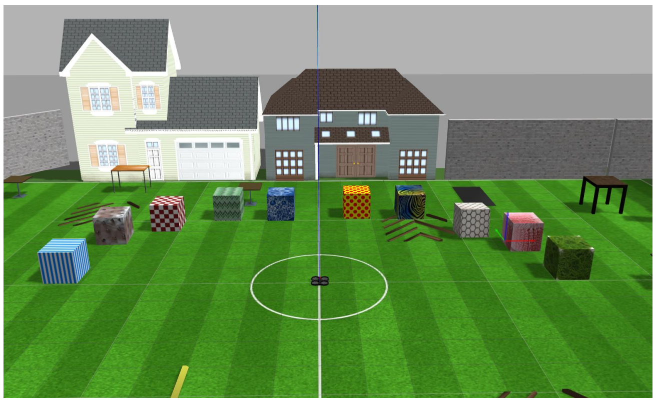


Figure 1: We designed a system for texture classification in aerial navigation based on knowledge inference over the DTD database. See at `https://youtu.be/d41kgBw7Y_c`.

2 RELATED WORK

In recent years, MAVs applications for object detection tasks have been studied and developed [6][7]. Several approaches are using deep learning, giving excellent performance in applications. For example, in some tasks for autonomous navigation, we find in the literature a methodology for obstacle detection and avoidance using an architecture called AlexNet that allows classifying the images captured by the camera onboard the drone [8]. The learning of this architecture is transferred from the ImageNet database to improve the classification performance [9]. Moreover, to detect objects and autonomous landing, in [1], a detection system is presented to solve one of the missions included in the IMAV2019 indoor competition. They involve the implementation of the SSD7 onboard the MAV. This SSD7 network is chosen for its fast performance on low-budget microcomputers with no GPU. The method proposed in [10] is an architecture called YOLO, which presents an essential performance in real-time image detection and processing at 45 frames per second. Besides, in object detection tasks, in [11], the authors propose a deep learning approach to estimate the object's center in a robust way. Also, generating a line of sight as a guide proves to be a solution to avoid collision with other objects, due to complications such as varying illumination conditions, object geometry, and overlapping in the image plane.

On the other hand, many projects employ deep learning and wavelet analysis in visual processing. For example, on image classification, the method proposed in [12] converts images from the CIFAR-10 and KDEF database to the wavelet domain, thus obtaining temporal and frequency features. The different representations are added to multiple CNN architectures. This combination of information in the wavelet domain achieves higher detection efficiency and faster execution times than the spatial domain procedure. In this sense, the authors in [13] mention that although CNN is a universal extractor, in practice, it is not clear whether CNN can learn to perform spectral analysis. In [14], the authors propose an architecture called CNN Texture to have this approach within the CNN. Their idea focuses on the fact that the information extracted by convolutional layers is of minor importance in texture analysis. They use a statistical energy metric in the feature extraction stage. This information is concatenated with the classification stage, the fully connected layer. Specifically, the architecture shows an improvement in performance and a reduction in computational cost.

In terms of texture classification in image processing applications [15], the authors propose an architecture called wavelet CNN to generalize spectral information lost in conventional CNNs. This information is beneficial for texture classification as it usually contains details information about the object's shape. Furthermore, the model allows us to have fewer parameters than in traditional CNNs, so it is possible to train with less memory. In general, through a state-of-the-art review, we have observed that computational intelligence algorithms improve detection strategies in Micro Aerial Vehicle applications.

3 METHODOLOGY

This work proposes an approach based on transfer learning and wavelet features. This system allows to predict or classify the texture in images transmitted by the camera onboard the drone, whose objects are in an outdoor scenario (in Gazebo), a virtual simulation environment. We are only interested in texture recognition, mainly to know one of the characteristics of the object. So, we limit the image plane (640×360) to a region of interest (300×300 pixels). As a result, the system will have the image in RGB as well as a grayscale version. These two images are the inputs for our proposed classification system, see figure 2.

Describable Textures Dataset (DTD) was selected to be used. It contains 47 classes of 120 images in the wild. This means that the images have been acquired that in uncontrolled

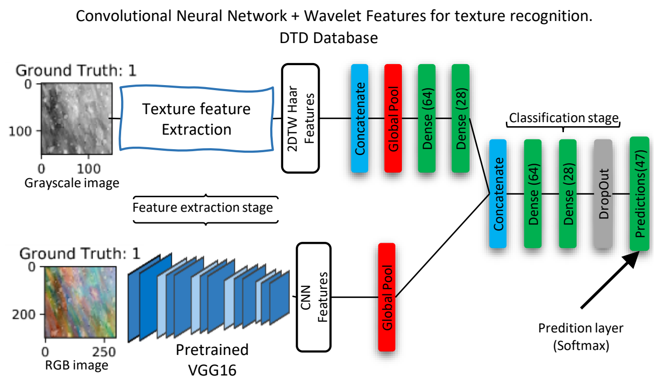


Figure 2: Texture classification system.

conditions [16]. This dataset includes ten divisions available with 40 training images, 40 validation images, and 40 test images for each class. Our experiment will create a new dataset, with the distribution of 70% for training, 15% for validation, and the remaining 15% for testing. Figure 3 shows some images from this set. One limitation of the dataset is the number of images per class, so it is decided to use the transfer learning method to improve the classification performance of our model. The synaptic weights are based on the ImageNet database, which will feed the feature extraction stage of the base architecture VGG16.

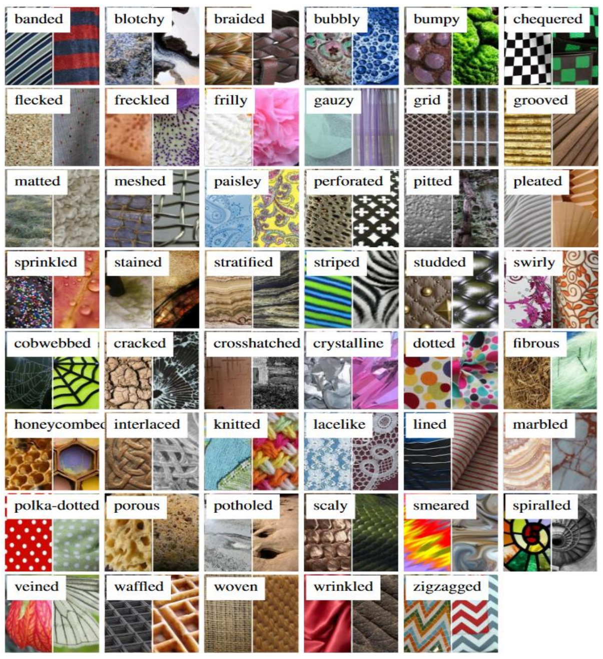


Figure 3: DTD dataset example images [16].

Before training, the Haar wavelet transforms in two dimensions is applied to one level, see figure 2. The factor of one represents the level of image decomposition. This new spectral information is essential for classification. Therefore, four sets (in the wavelet domain) are automatically generated to determine the characteristic attributes of each texture. This information can be combined with the spatial information of the VGG16 architecture. Also, it is essential to mention that this process is only applied to the image previously converted to grayscale, performing the decomposition for a single channel, ver figure 4 & 5.

For the test stage in MAV, a scenario is designed in the gazebo simulator. The virtual scenario is created with ten cubes with certain textures (figure 1). These textures are selected due to the performance achieved in the model testing stage. Therefore, the chosen classes have a performance above 70% accuracy (Table 3).

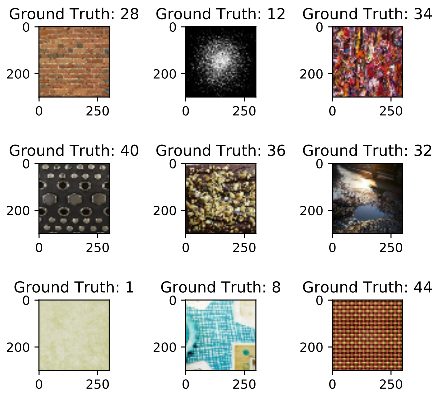


Figure 4: Images textures that have been decoded (Class) to train the classification model.

4 EXPERIMENTS AND RESULTS

The experiment to train our learning model was carried out with the Keras API with Tensorflow as Backend [17]. Besides, the OpenCV libraries are used for image processing due to their ease of use and adaptability in programming. Also, we use the Pywt library [18] from which it was chosen the Haar wavelet transform as the feature extractor method. An aerial navigation experiment was performed using the ROS framework and Gazebo simulation environment to validate the classification system and its learning generalization. This section describes the results obtained in each experiment.

4.1 Model training

In the first instance, the VGG16 network was trained from scratch. As a result, it is not able to generalize its learning. Therefore, it is possible to use the transfer learning methodology. Table 1 shows the achieved performance of the pre-trained network and our proposal with the wavelet feature fusion. It shows the accuracy performance on the three sets to validate the model (training, validation, and test). In the case of the pre-trained network, slight overfitting is observed. The model will be adjusted to learn specific cases and will be

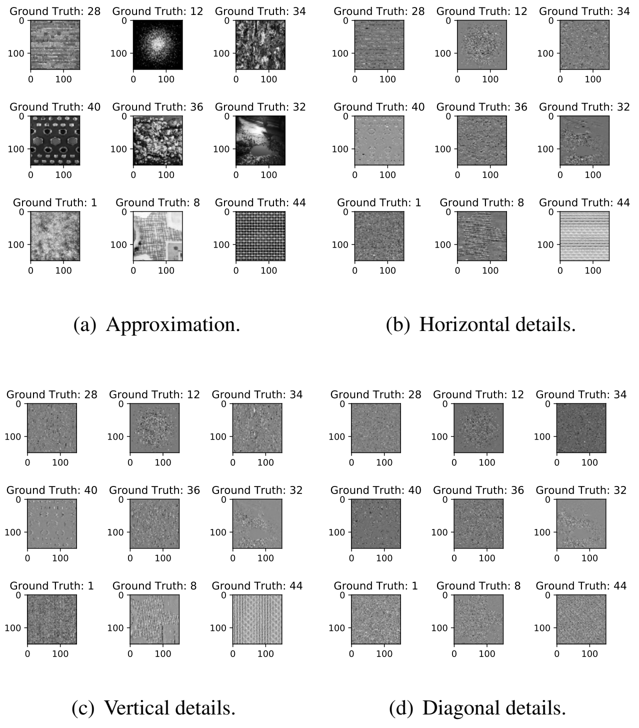


Figure 5: Approximation and details set of wavelet features.

unable to recognize new textures. One way to improve the performance of the model is to integrate the wavelet features. In this case, we achieve the elimination of overfitting and homogeneity between the three sets. Besides, the value of the test set is highlighted because these are images that the model has never seen. In summary, the classification system has 14,778,735 synaptic learning weights. 64,047 are trainable parameters, of which 16,832 correspond to wavelet features. Table 2 summarizes the achieved performance of our classification system, as well as a comparison to AlexNet (trained from scratch), T-CNN, and Wavelet CNN [14][15].

	Training	Validation	Test
Pre-trained model	68.15	50.41	54.49
Our model	57.67	51.22	**53.19**

Table 1: Classification results for the pre-trained VGG16 network and our model indicated as accuracy (%).

	AlexNet	T-CNN	Wavelet CNN	Our model
DTD	22.7	55.8	59.8	**53.19**

Table 2: Classification results and comparison with other state-of-the-art pre-trained architectures with ImageNet, in terms of accuracy (%).

4.2 *Texture classification DTD*

Other metrics evaluate the performance of the DTD dataset classes. The metrics such as precision, recall, and f1-score are given when applying the classification_report method, where it is necessary to involve the true labels and the prediction label of the model. Table 3 shows the classes that performed above 70% classification. Also, Table 4 shows three random classes that perform above 50% classification. This class selection analysis provides the basis for the design of the textured cubes of the Gazebo environment. On the other hand, we can observe the similarity and correlation between classes (about test set) by performing the prediction. Figure 6 shows the true label and the prediction label at the top of each texture.

Class	precision	recall	f1-score	support
bubb	0.73	0.61	0.67	18
cheq	1.00	0.78	0.88	18
fibr	0.73	0.61	0.67	18
fril	0.72	0.72	0.72	18
stri	0.77	0.94	0.85	18
stud	0.70	0.78	0.74	18
zigz	0.75	0.67	0.71	18

Table 3: Classes (test set) that results with precision above 70%.

Class	precision	recall	f1-score	support
hone	0.58	0.61	0.59	18
line	0.50	0.28	0.36	18
polk	0.65	0.61	0.63	18

Table 4: Classes (test set) that results with precision above 50%. They are chosen from the easy human visual perception of the texture.

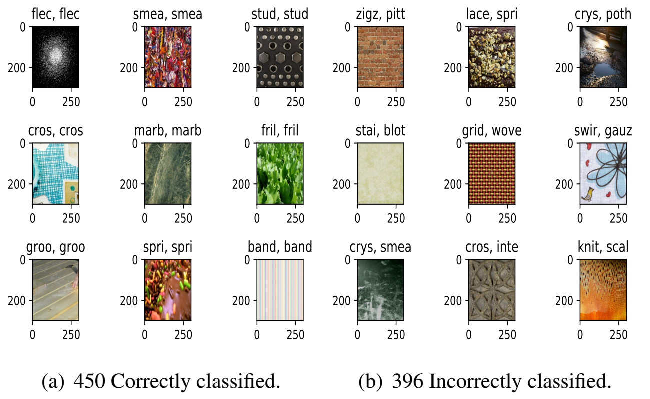


Figure 6: Classification of textures randomly (from a total of 846 images) using the DTD prediction model.

4.3 *Texture classification in aerial navigation*

Navigation and aerial recognition tested the classification model. We created a virtual environment with the Gazebo simulator, controlling and sending information from the camera onboard the drone using the ROS framework. In the world presented in figure 1, we positioned in a row the ten cubes with the selected textures. Therefore, the position of the cubes allows the evaluation of the prediction model during aerial exploration. The idea of the model is that it generalizes its learning to textured objects. In total, 1000 image captures were performed in a navigation recognition for each class. The proposed texture sets (bubbly, chequered, honey, striped, studded) obtain a high correlation with their original label above 60% accuracy, see figure 7.

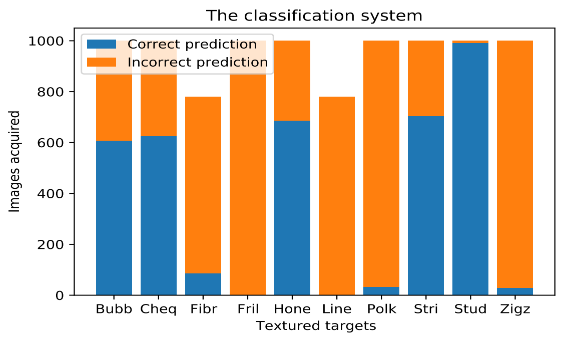


Figure 7: The number of images with textures obtained with the onboard camera while flying recognition.

Some images (figures 8 and 9) of the recognition set are shown, with its original label and its prediction label. However, we can observe that the five test images incorrectly predict the frilly, lined, polka-dotted, and zigzagged set. These five images relate to the whole recognition set, except for fibrous, polka-dotted, and zigzagged, which achieve at least 3% accuracy. This result allows us to understand the generalization of learning between the model and textured objects.

5 Conclusion

The localization and object detection tasks using visual information are challenging, particularly when objects exhibit repetitive texture. However, these tasks open the opportunity for various applications using Micro Aerial Vehicles equipped with onboard cameras to be used for object detection and recognition, for instance, for parcel pick-up, place recognition, landing zone detection and many more. Seeking to improve the detection and recognition stage, in this work we have investigated the use of spectral analysis in combination with deep neural networks. In particular, in this proposal, we merged the (additionally created) spectral feature maps to CNN learning. Also, it is shown that the model used achieves to eliminate overfitting and better accuracy in the classification of textures with a significant increase in the number of

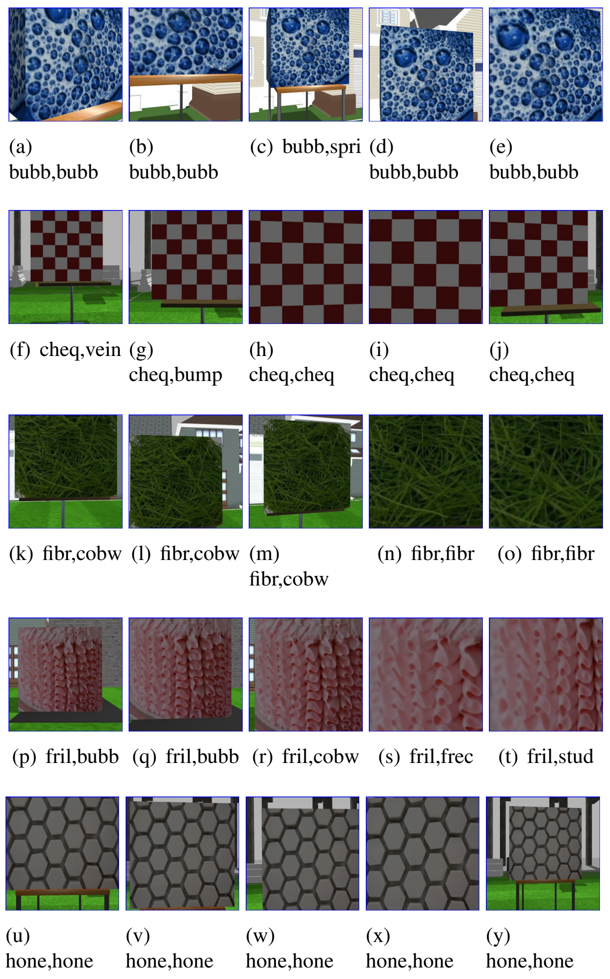


Figure 8: Image sequence acquired by the camera onboard the drone. The classification system has a good inference on the texture in the first, second, and fifth rows.

parameters to be trained. The tests performed in the simulation show some interesting results. The prediction model shows the creation of a widespread understanding of the texture attached to the objects. Furthermore, despite having a low classification rate, it is shown that the model correctly classifies most of the test classes.

As future work, this will test with other texture features, also seeking to conduct tests in real-world scenarios.

Acknowledgements

J. M. Fortuna-Cervantes is a doctoral fellow of CONACYT (México) in the program of “Ciencias Aplicadas” at UASLP-IICO.

References

[1] Aldrich A Cabrera-Ponce and José Martínez-Carranza. Onboard cnn-based processing for target detection and autonomous landing for mavs. In *Mexican Conference on Pattern Recognition*, pages 195–208. Springer, 2020.

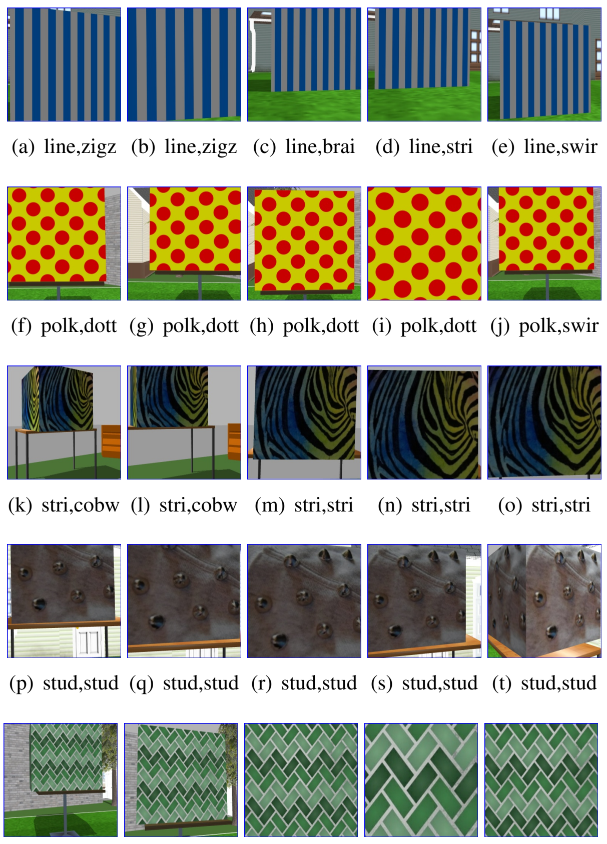


Figure 9: Image sequence acquired by the camera onboard the drone. In the second, third, and fourth rows, the classification system gets good classification performance.

[2] Natalia S Vassilieva. Content-based image retrieval methods. *Programming and Computer Software*, 35(3):158–180, 2009.

[3] LC Yan, B Yoshua, and H Geoffrey. Deep learning. *nature*, 521(7553):436–444, 2015.

[4] Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep learning*, volume 1. MIT press Massachusetts, USA:, 2017.

[5] Stephane G Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE transactions on pattern analysis and machine intelligence*, 11(7):674–693, 1989.

[6] Yakoub Bazi and Farid Melgani. Convolutional svm networks for object detection in uav imagery. *Ieee transactions on geoscience and remote sensing*, 56(6):3107–3118, 2018.

[7] Yalong Pi, Nipun D Nath, and Amir H Behzadan. Convolutional neural networks for object detection in aerial imagery for disaster response and recovery. *Advanced Engineering Informatics*, 43:101009, 2020.

[8] Sinahi Dionisio-Ortega, L Oyuki Rojas-Perez, Jose Martinez-Carranza, and Israel Cruz-Vega. A deep learning approach towards autonomous flight in forest environments. In *2018 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, pages 139–144. IEEE, 2018.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[10] 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.

[11] Sunggoo Jung, Sunyou Hwang, Heemin Shin, and David Hyunchul Shim. Perception, guidance, and navigation for indoor autonomous drone racing using deep learning. *IEEE Robotics and Automation Letters*, 3(3):2539–2544, 2018.

[12] Travis Williams, Robert Li, et al. An ensemble of convolutional neural networks using wavelets for image classification. *Journal of Software Engineering and Applications*, 11(02):69, 2018.

[13] Travis Williams, Robert Li, et al. Wavelet pooling for convolutional neural networks. *Proc. Int. Conf. on Learning Representations*, 2018.

[14] Vincent Andrearczyk and Paul F Whelan. Using filter banks in convolutional neural networks for texture classification. *Pattern Recognition Letters*, 84:63–69, 2016.

[15] Shin Fujieda, Kohei Takayama, and Toshiya Hachisuka. Wavelet convolutional neural networks. *arXiv preprint arXiv:1805.08620*, 2018.

[16] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3606–3613, 2014.

[17] Francois Chollet et al. *Deep learning with Python*, volume 361. Manning New York, 2018.

[18] G. Lee, K. Wohlfahrt R. Gommers, F. Waselewski, and A. O'Leary. Pywavelets: A python package for wavelet analysis. In *Journal of Open Source Software*, volume 4, page 1237, 2019.

Bibliography

[1] Dennis van Wijngaarden, Ewoud Smeur, and Bart Remes. Flight code convergence: Fixedwing, rotorcraft, hybrid. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 21–27, Puebla, México, Nov 2021. Paper no. IMAV2021-1. 16

[2] Shushuai Li, Christophe De Wagter, and Guido de Croon. Nonlinear model predictive control for improving range-based relative localization by maximizing observability. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 28–34, Puebla, México, Nov 2021. Paper no. IMAV2021-2. 16, 17

[3] Mathias Bos, Bart Theys, Jan Swevers, and Goele Pipeleers. Modeling and identification of multirotor drone dynamics for onboard mpc motion planning. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 35–41, Puebla, México, Nov 2021. Paper no. IMAV2021-3. 16

[4] Erik Vroon, Jim Rojer, and Guido de Croon. Motion-based mav detection in gps-denied environments. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 42–49, Puebla, México, Nov 2021. Paper no. IMAV2021-4. 16

[5] Miguel Fernandez-Cortizas, Pablo Santamaría, David Perez-Saura, Javier Rodríguez-Vázquez, Martin Molina, and Pascual Campoy. Framework and evaluation methodology for autonomous drone racing. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 50–56, Puebla, México, Nov 2021. Paper no. IMAV2021-5. 16, 17

[6] Daniel Olivares-Figueroa, Israel Cruz-Vega, Juan Manuel Ramírez-Cortes, María del Pilar Gómez-Gil, and José Martínez-Carranza. A compact approach for emotional assessment of drone pilots using bci. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 57–62, Puebla, México, Nov 2021. Paper no. IMAV2021-6. 16

[7] Hector Garcia de Marina, Murat Bronz, and Gautier Hattenberger. Guiding vector fields in paparazzi autopilot. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 63–67, Puebla, México, Nov 2021. Paper no. IMAV2021-7. 16

[8] Aaron Lopez, Hugo Rodríguez Cortés, Israel Cruz Vega, and José Martínez-Carranza. Immersion and invariance based trajectory tracking control of an aerial manipulation system. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 68–73, Puebla, México, Nov 2021. Paper no. IMAV2021-8. 16, 17

[9] Jan Karssies and Christophe De Wagter. Extended incremental non-linear control allocation on the tu delft quadplane. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 74–84, Puebla, México, Nov 2021. Paper no. IMAV2021-9. 16, 17

[10] Guillermo Gonzalez, Guido de Croon, Diana Olejnik, and Mat?j Karázek. Position controller for a flapping wing drone using uwb. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 85–92, Puebla, México, Nov 2021. Paper no. IMAV2021-10. 16, 17

[11] Jelle Westenberger, Christophe De Wagter, and Guido de Croon. Onboard time-optimal control for tiny quadcopters. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 93–100, Puebla, México, Nov 2021. Paper no. IMAV2021-11. 16, 17

[12] Rmain Jan, Jean-Philippe Condomines, and Jean-Marc Moschetta. Fast simulation model for control law design andbenchmark of high aspect ratio flexible uavs. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 101–108, Puebla, México, Nov 2021. Paper no. IMAV2021-12. 16

[13] Gautier Hattenberger, Titouan Verdu, Nicolas Maury, Pierre Narvor, Fleur Couvreux, Murat Bronz, Simon Lacroix, Grégoire Cayez, and Gregory Roberts. Field report: deployment of a fleet of drones for cloud exploration. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 109–115, Puebla, México, Nov 2021. Paper no. IMAV2021-13. 16, 17

[14] Midas Gossye, Sunyou Hwang, and Bart Remes. Developing a modular tool to simulate regeneration power potential using orographic wind-hovering uavs. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 116–123, Puebla, México, Nov 2021. Paper no. IMAV2021-14. 16, 17

[15] Gautier Hattenberger, Murat Bronz, and Jean-Philippe Condomines. Estimating wind using a quadrotor. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 124–130, Puebla, México, Nov 2021. Paper no. IMAV2021-15. 16, 17

[16] Lars Dellemann and Christophe De Wagter. Hybrid uav attitude control using indi and dynamic tilt-twist. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 131–136, Puebla, México, Nov 2021. Paper no. IMAV2021-16. 16

[17] Ivan Martínez Pérez, Rodolfo García Rodríguez, Mario Alejandro Vega Navarrete, and Luis Enrique Ramos Velasco. ” sliding-mode based thrust vector control for aircrafts”. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 137–143, Puebla, México, Nov 2021. Paper no. IMAV2021-17. 16

[18] Juan Ayala, Rogerio Enriquez, and Fermi Guerrero. Quaternion based attitude sliding mode control with disturbance rejection observer for a quadrotor. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 144–152, Puebla, México, Nov 2021. Paper no. IMAV2021-18. 16

[19] Pietro Li Volsi, David Gomez-Ariza, Thierry Jardin, Romain Gojon, and Jean-Marc Moschetta. Design of aeroacoustically stealth mav rotors. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 153–160, Puebla, México, Nov 2021. Paper no. IMAV2021-19. 16, 17

[20] Shang Lin Wu, Yao Wei Chin, and Gih Keong Lau. A click mechanism moderates drone’s flapping wing kinematics for enhanced thrust generation. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 161–165, Puebla, México, Nov 2021. Paper no. IMAV2021-20. 16

[21] Liam Bullard, Abdulghani Mohamed, and Simon Watkins. Propulsive efficiency of small multirotor propellers in fast forward flight. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 166–171, Puebla, México, Nov 2021. Paper no. IMAV2021-21. 16

[22] Nikola Gavrilovic, Phassawat Leelaburanathanakul, Javier Cuadrado-Anibarro, and Jean-Marc Moschetta. Aeropropulsive performance improvement of h2 powered uas. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 172–179, Puebla, México, Nov 2021. Paper no. IMAV2021-22. 16

[23] Sergey Serokhvostov and Tatiana Churkina. Implementation of copter propeller model to the problem of energy consumption minimization during lift phase. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 180–187, Puebla, México, Nov 2021. Paper no. IMAV2021-23. 16

[24] Carlos Alexander Osorio Quero, Daniel Durini Romero Durini Romero, Jose Rangel-Magdaleno, José Martínez-Carranza, and Ruben Ramos-Garcia. 3d reconstruction based on nir single-pixel for drone navigation under rainy condition. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 188–195, Puebla, México, Nov 2021. Paper no. IMAV2021-24. 16

[25] Andrés Solares Jurado, Germán Andrés Di Fonzo, Rafael Pérez, Hriday Bavle, Miguel Fernandez-Cortizas, Javier Rodríguez-Vázquez, Guillermo Robledo, and Pascual Campoy. Indoor visual semantic slam improves vio and rgbd for narrow space navigation. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 196–203, Puebla, México, Nov 2021. Paper no. IMAV2021-25. 16

[26] Xinyi Wang, Lele Xi, Yizhou Chen, Shupeng Lai, Feng Lin, and Ben M. Chen. Decentralized trajectory generation technique for multiple unmanned multicopter systems in cluttered environments. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 204–209, Puebla, México, Nov 2021. Paper no. IMAV2021-26. 16

[27] Juan Manuel Fortuna-Cervantes, Marco Tulio Ramírez-Torres, Marcela Mejía-Carlos, José Martínez-Carranza, and José Salomé Murguía-Ibarra. Texture classification for object detection in aerial navigation using transfer learning and wavelet-based features. In Jose Martinez-Carranza, editor, *12th International Micro Air Vehicle Conference*, pages 210–215, Puebla, México, Nov 2021. Paper no. IMAV2021-27. 16